Universidade de Aveiro
Ano 2014

Departamento de Electrónica,
Telecomunicações e Informática

Tiago José da Fonseca Marques

STRIVE: Partilha de informação de trânsito citadino com Smartphones

STRIVE:using Smartphone to share TRaffic Information in citywide VanEt

Universidade de Aveiro
Ano 2014

Departamento de Electrónica,
Telecomunicações e Informática

Tiago José da Fonseca Marques

STRIVE: Partilha de informação de trânsito citadino com Smartphones

STRIVE:using Smartphone to share TRaffic Information in citywide VanEt

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor José Maria Amaral Fernandes e da Doutora Susana Sargento, Professores Auxiliares do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais e família pelo incansável apoio.

O júri

Presidente

Prof. Doutor Rui Luís Andrade Aguiar
Professor associado c/Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Vogais

Prof. Doutor José Maria Amaral Fernandes
Professor auxiliar no Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro. (Orientador)

Prof. Doutor Pedro Miguel Alves Brandão
Professor auxiliar no Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto. (Arguente Principal)

Agradecimentos

Quero agradecer em primeiro lugar aos meus Pais por me proporcionarem a oportunidade de frequentar um curso superior e me apoiarem sempre durante o meu percurso académico facilitando-me sempre a melhor ajuda possível.

Quero agradecer a todos os colaboradores do Instituto de Telecomunicações que me ajudaram durante o meu trabalho e se mostraram sempre disponíveis. Um agradecimento especial para o Bruno Areias pela paciência e ajuda prestada na integração do meu trabalho com o equipamento.

Agradeço a todos os meus colegas de curso e amigos pelo apoio e motivação que prestaram sempre. Em especial agradecer ao Henrique Costa e ao José Mendonça que apesar de não estarem envolvidos no desenvolvimento do trabalho acompanharam de perto todo o processo e foram muito importantes no meu percurso académico.

Por último, mas não menos importante, quero agradecer à Professora Susana Sargento e ao Professor José Maria Fernandes pela orientação, coordenação e dedicação prestada durante todo o trabalho, transmitindo-me sempre uma sensação de confiança e segurança para o trabalho.

palavras-chave

Android, Aplicação, Redes veiculares, *smartphones*, VANET, REST, Content Provider, Serviços.

Resumo

As redes veiculares têm sofrido diversos avanços nos últimos anos. A comunicação entre veículos é cada vez mais uma realidade no mundo actual e tem sido alvo de investigação e desenvolvimento quer por investigadores, quer pelos principais fabricantes de automóveis, com o objectivo de trazer até aos condutores diversos serviços, usando as redes veiculares como meio de transporte, através do uso dos seus *smartphones*. Numa primeira avaliação, a principal vantagem da rede veicular assenta na troca de mensagens entre veículos que permita criar um sistema de alertas para os condutores em situações de perigo, tentando reduzir o número de acidentes nas estradas. Ao mesmo tempo, nos últimos anos tem-se assistido a um crescimento constante da capacidade de processamento e de funcionalidades dos equipamentos móveis ao mesmo tempo que se assiste ao aumento das suas vendas. Os dispositivos móveis como *smartphones* e *tablets* têm tido uma grande proliferação, e cada vez mais se assiste a um uso das mais variadas aplicações por parte de qualquer utilizador. Desta forma, existe além da questão da segurança uma tentativa de implementar o uso das capacidades das redes veiculares em novas aplicações que tragam até ao utilizador, seja ele condutor ou passageiro, serviços que vão além da segurança, como comunicação e entretenimento. Estes serviços permitirão fazer uso do elevado poder computacional que os *smartphones* disponibilizam e que não se encontram mais desenvolvidos devido à falta de um sistema completo que inclua uma solução capaz de garantir conectividade total entre os diversos intervenientes, desde os componentes intrínsecos à rede veicular até à camada de aplicação dos *smartphones* da actualidade.

Este trabalho propõe-se a estudar, arquitectar e desenvolver o STRIVE, uma solução que pretende resolver os problemas de comunicação e conectividade entre redes móveis e aplicações de alto nível, incluindo também a possibilidade de obter informação directamente do veículo através de um módulo de ligação ao sistema de diagnóstico. Assim, os cenários possíveis incluem tráfego de dados de vários tipos como recursos web, rede veicular e sensores. Para agregar e processar estes dados foi desenvolvida uma aplicação móvel que fornece funcionalidades ao condutor, como um sistema de alertas de perigos na estrada, alertas de problemas no veículo, relatórios de rotas e consumos ou comunicação textual entre veículos.

No âmbito da Dissertação foram criados alguns cenários de testes para análise de desempenho da aplicação e das comunicações em ambiente real e simulado. Estes testes serviram para identificar a viabilidade do STRIVE em dispositivos reais, usando redes de comunicação reais. Deste modo pretende-se também observar os possíveis problemas do sistema.

Os resultados obtidos permitiram constatar que o sistema pode ser utilizado por qualquer proprietário de um *smartphone* Android, que faça uso da rede veicular testada, apenas instalando a aplicação desenvolvida.

Keywords

Android, Application, Vehicular communication, *smartphones*, VANET, REST, Content Provider, Services.

Abstract

In recent years, vehicular networks have undergone many advances. Communication between vehicles is increasingly becoming a reality in today's world and has been the target in R&D either by researchers or by major car manufacturers, with the goal of bringing up various services to the drivers, using the vehicular networks as means of transport, through the use of their *smartphones*.

In a first review, the main advantage of the vehicular network is based on the exchange of messages between vehicles allowing the creation of an alert system for drivers in a dangerous situation, in order to try to reduce the number of road accidents.

At the same time, in the last few years, there has been a steady growth in the processing power and functionality of mobile devices and an increase in their sales. Mobile devices, such as *smartphones* and tablets, have come here to stay and, increasingly, we are witnessing a more varied use of applications by the users.

Thus, there is, beyond the issue of security, an attempt to implement the use of the capabilities of vehicular networks in new applications that bring to the user, be it driver or passenger, services that go beyond security, such as communication and entertainment. These services will make use of the high computational power that *smartphones* offer, and that are not more developed due to the lack of a complete system that includes a solution capable of full connectivity between the various "actors", from the intrinsic components, to the vehicular network and to the application layer of the *smartphones* of today.

This work in this Dissertation aims to study, architect and develop a solution that solves the problems of communication and connectivity between vehicular networks and high-level applications, including also the possibility of obtaining information directly from the vehicle via a connection module to the car diagnostic's system.

So, the possible scenarios include traffic data of various types such as: Internet, vehicular network and sensors. To aggregate and process this data, it was developed, as an end product of the system, a mobile application that provides functionality to the driver as a system of alerts for road hazards, warnings concerning problems in the vehicle, route and consumption reports or textual communication between vehicles.

Within this Master's Dissertation, several test scenarios for application and communications performance, in real and simulated environment, were created. These tests served to identify the feasibility of the STRIVE on real devices, using real communication networks. Thus, the aim was also to observe the possible problems of the system.

The results obtained allowed us to prove that the system can be used by any owner of an Android smartphone, who makes use of the tested vehicular network, just by installing the developed application.

Table of Contents

Lista de figuras

Lista de tabelas

1.Introdução

A integração de novas tecnologias nos veículos motorizados é uma realidade. Bons exemplos são o caso do GPS, ESP (Electronic Stability Control) que ajuda no sistema de tracção, sistema de apoio a travagem, sistemas de leitura dos sinais da estrada, sistemas de alerta de trânsito intenso, entre outros. Todos estes sistemas envolvem novas tecnologias e inovação.

No entanto, só mais recentemente a existência de conectividade entre veículos e entre veículos e infra-estruturas fixas que permitam ligar o carro à Internet de modo eficiente, começou a ser uma realidade [1]. No entanto, muitas das soluções que alguns fabricantes de automóveis disponibilizam para este efeito são de custos muito elevados. Estas soluções traduzem-se quer numa perspectiva de aplicações para ajudar o condutor no dia-a-dia, quer, por exemplo, numa perspectiva empresarial como é o auxílio da gestão e controlo de frotas de empresas variadas como táxis, autocarros, camiões, que pode ajudar na redução de custos com combustível e tempo aumentando as vantagens económicas.

O *European Telecommunications Standard Institute* defende que o *standard* DSRC (*Dedicated Short Range Communications*) [2] é uma solução viável para comunicações entre veículos e que poderá servir de suporte à implementação de serviços/aplicações básicas, para melhorar os sistemas inteligentes de transportes, *Intelligent Transport System* (ITS) [3], nomeadamente em cenários de segurança rodoviária [4].

Contudo é preciso ainda realçar que as redes veiculares que permitem a comunicação entre veículos, VANET (Vehicular Ad-hoc NetWork), utilizam redes *wireless* de *standards* não suportados directamente pelos telemóveis. Assim, é necessária a instalação nos veículos de equipamentos com capacidade de emitir e receber dados em comunicações de protocolo WAVE, composto pelos *standards* IEEE 802.11p e a família 1609.x.

Em simultâneo, nos últimos anos temos assistido a um aumento muito grande da investigação e criação de novas tecnologias como a nanotecnologia, que permitiu que se desenvolvesse em tamanho reduzido, componentes com grande capacidade de armazenamento e/ou processamento. Isto traduz-se em pequenos telemóveis com elevado poder computacional, vulgarmente designados de *smartphones*.

A par destas inovações no *hardware* foram criados sistemas operativos novos e optimizados para *smartphones* como o Android, iOS e Windows Phone, tornando assim o desenvolvimento de aplicações mais simples.

Assim, os telemóveis podem servir de base para criar essas aplicações para as várias áreas das VANETs [5], e criar um conjunto de serviços, desde comunicação entre ocupantes de diferentes veículos, partilha de informação rodoviária sobre trânsito, estado das estradas até à partilha de informação sobre cada veículo entre outros, que sejam uma grande mais-valia ao condutor e/ou passageiros.

1.2 Motivação

No caso das redes veiculares, estão disponíveis os módulos principais para que se possa criar um sistema completo que traga as mais diversas vantagens ao utilizador. Os componentes que permitem estabelecer a ligação já existem, embora de forma ainda muito limitada, assim como *smartphones* de elevada capacidade computacional.

No entanto, ainda não existem soluções que permitem aproveitar as mais-valias de juntar estes dois sistemas, desde a segurança passiva na condução até ao simples entretenimento para passageiros.

Este foi um dos pontos de motivação para este trabalho: conectar estes dois sistemas independentes, VANET e telemóveis, de modo a criar o suporte para que as redes veiculares estejam disponíveis ao nível aplicacional.

Dada a quantidade de informação disponível e o grande potencial que as comunicações entre veículos fornecem, e tendo em conta a evolução tecnológica presente hoje nos telemóveis da maioria dos cidadãos, torna-se importante desenvolver sistemas com aplicações móveis como *front-end* que utilizem e tratem estes dados para mostrar resultados ao utilizador.

Segundo estudos da Cisco [6], o tráfego de dados por telemóvel evoluiu cerca de 81% em 2013, o que demonstra um acréscimo de uso por parte das pessoas para aceder à Internet e usar as diversas potencialidades dos seus aparelhos.

Outra realidade que demonstra isso é o número crescente de aplicações nas duas principais plataformas iOS (App Store) [7] e Android (Play Store) [8], pertencentes respectivamente à Apple e Google, que mostra o interesse económico nas receitas que esta evolução das tendências pode trazer.

O mercado de *smartphones* Android é o que mais tem crescido desde 2009 até ao presente ano de 2014, e os estudos mostram que esta tendência se irá manter. Assim, justifica-se a escolha desta plataforma para uma primeira fase de desenvolvimento e teste das novas aplicações.

Tem-se verificado nos últimos anos um acréscimo na evolução no sector automóvel dos veículos híbridos no sentido de diminuir o consumo dos mesmos e reduzir custos ao condutor. Esses custos podem ainda ser diminuídos se recorrermos a ferramentas que permitam fazer análise do consumo imediato do veículo, análise do trânsito nas rotas pretendidas, monitorização de gastos em frotas de veículos de empresas entre outras possibilidades.

Com as principais aplicações e alguns casos de uso definidos, e com as redes veiculares disponíveis, este trabalho permitirá evoluir os sistemas utilizadores de redes veiculares para que estas estejam disponíveis ao nível aplicacional, tendo impacto directo no utilizador.

1.3 Objectivos

O objectivo geral desta Dissertação é criar uma solução que combine os novos *smartphones* do mercado e as redes veiculares. Esta combinação pode-se traduzir em aplicações de diferentes tipos [5], que podem servir de auxiliares à condução para questões de consumo, gestão de viagens e percursos de curta ou longa distância, manutenção de veículos, sistemas de alertas em tempo real sobre possíveis perigos entre outras aplicações. Deste modo, surge a aplicação *MyCar*, que traz estas e outras vantagens ao utilizador comum.

Além da aplicação, o sistema engloba serviços *web* remotos para acesso, quer à base de dados comum a todas as aplicações, quer a serviços externos para obtenção de mapas e locais de interesse. Através do acesso à base de dados é possível a criação de um *web-site* para visualização dos dados disponíveis em cada instante e em qualquer lugar, aumentando desta forma o cenário para o âmbito empresarial como a gestão de frotas.

Além da criação das ferramentas para ajudar ao desenvolvimento de sistemas que utilizem as redes veiculares, pretende-se também desenvolver um cenário para mostrar a viabilidade e importância que estes sistemas podem ter no melhoramento da condução.

1.4 Contribuição

A contribuição deste trabalho centra-se no desenvolvimento de um novo sistema – **STRIVE**: Partilha de informação de trânsito citadino com *smartphones* – que permite que as comunicações na rede veicular estejam disponíveis em alto nível para uso em aplicações de utilizadores. Num contexto em que temos VANETs e equipamentos *mobile* faz sentido que os serviços e troca de informação sejam mais simples. Actualmente grande parte das soluções que integram VANETs implicam lidar directamente com o nível de transporte. Além disso, apesar da relevância dada aos serviços sobre ITS, estes ainda não são suportados ao nível do utilizador / aplicação, apesar de alguns cenários fazerem sentido, como por exemplo, acesso aos dados dos automóveis pelo utilizador / gestor de frota, acesso a informação do contexto rodoviário (acidentes, problemas no pavimento, congestionamento de trânsito) e de cariz mais social e entretenimento (onde estão "amigos", colegas ou outros utilizadores relevantes).

Assim houve duas contribuições principais neste trabalho que merecem um destaque diferenciado:

- Elaboração de um cenário envolvendo redes veiculares e os requisitos enumerados previamente, e criação de um sistema, com uma aplicação Android incluída, a aplicação *MyCar*, que visa responder a todas as necessidades que o cenário requer.
- Melhoramento e ampliação da camada de abstracção para a camada de transporte usada como base de trabalho na construção de uma nova API. Esta API tem como objectivo

permitir a qualquer aplicação o uso da rede veicular como camada de transporte de forma simples e transparente.

Por um lado foi elaborado um cenário especificado através de análises de alguns casos de uso comuns a várias aplicações [5] que se baseiam em VANETs. Esses casos de uso incluem desde os mais simples, como o envio de mensagens de texto para os veículos em redor, até aos mais complexos como reencaminhar um pedido *web*, de outro veículo, para a Internet ou mesmo aceder à base de dados remota para visualizar dados. O cenário, posteriormente descrito em pormenor, vai evidenciar os vários casos de uso explorados e as várias áreas de aplicação que o cenário pode abranger.

De acordo com o segundo objectivo, é implementada uma camada de abstracção para lidar com a camada de transporte na comunicação WAVE e fornecer suporte a todas as funcionalidades da aplicação *MyCar*. Quando se fala em VANETs, falamos de ligações relativamente pouco exploradas no que toca a uma camada que converta as mensagens que queremos enviar para formatos usados na VANET (usualmente WAVE), e o inverso quando se pretende receber informação da VANET. Para servir de base ao objectivo de criar uma camada o mais genérica possível que sirva como API para outra qualquer aplicação poder transmitir e receber informação na VANET de forma simples e transparente, foi usado o trabalho desenvolvido no âmbito de dissertação de mestrado [9] que visava criar esta camada de abstracção entre camada de aplicação e de transporte. Contudo, apesar de funcional, este trabalho não servia para todos os objectivos deste projecto, tais como, envio de vários tipos de mensagens (alertas de mapa e de veículo, pedidos HTTP, valores fornecidos pelo veículo, mensagens de configuração) além dos já suportados ao nível de texto e GPS. Sendo assim, foi necessário estender as suas funcionalidades para garantir suporte à aplicação *MyCar*, fazendo todo o trabalho de conversão das mensagens que pretendemos enviar e receber.

A escolha de fazer este trabalho em Android prendeu-se com a facilidade de desenvolver para esta plataforma, e com o facto de ser um sistema largamente usado na actualidade [8]. Pretende-se que a API seja o mais genérica e simples possível, de modo a que qualquer utilizador possa desenvolver aplicações que usufruam das vantagens das redes veiculares sem qualquer preocupação de como se faz esse processo. O envio de mensagens do utilizador para a rede é feito com recurso a funções de envio fornecidas pela API. No caso da recepção de mensagens na VANET, é feito um alerta através de uma mensagem passada pela API a todas as aplicações que a usem a informar que estão disponíveis novas mensagens.

O sistema completo criado deve ser passível de ser usado por qualquer telemóvel que disponha de um sistema operativo Android. Contudo, devido ao uso de algumas funções recentes internas do sistema, aconselha-se que a versão do sistema seja 4 ou superior. O sistema inclui ainda alguns recursos remotos como base de dados e serviços *web*, que serão usados pela aplicação para lidar com funcionalidades específicas, como será explicado posteriormente neste documento.

1.5 Estrutura da dissertação

No capítulo 2 podemos encontrar o estado da arte, com contextualização das redes veiculares, influência das VANETS nos sistemas de transportes, principais áreas de aplicação das VANETS, terminando com projectos existentes que relacionam redes veiculares e ITS, e projectos que relacionam redes veiculares com aplicações móveis.

No capítulo 3 iremos introduzir o STRIVE, sistema que visa trazer até à camada de utilizador ou de aplicação as vantagens do meio de transporte das redes veiculares, através de uma nova API e uma aplicação *MyCar*. Iremos referir as principais funcionalidades que estão disponíveis usando o sistema.

Após essa introdução, irá ser abordado o REINVENT, plataforma existente para fornecer abstracção do meio veicular. Pode-se observar a arquitectura, o funcionamento baseado em REST, e o modo de comunicação por mensagens. Iremos também abordar um recurso externo usado, o RabbitMQ, um gestor de mensagens usado pelo REINVENT, e veremos como o RabbitMQ ajuda na conectividade entre os diferentes módulos do sistema. Por fim, neste capítulo iremos identificar os principais objectivos e cenários que o STRIVE tem de suportar.

No capítulo 4 serão apresentadas as implementações efectuadas no âmbito do STRIVE. O capítulo inicia com as implementações efectuadas em Android a nível de extensões do REINVENT para correcção de falhas, e para fornecer suporte às novas funcionalidades e novos tipos de mensagens criados. Serão abordadas as implementações realizadas nas OBUs dos veículos, os programas criados para permitir que a conectividade nas diferentes interfaces exista, recebendo, transformando e enviando mensagens entre as diferentes vias (WAVE, Wi-Fi, OBD). Vão ser também referidos os serviços *web* criados para acesso a uma base de dados central com dados de todos os veículos e condutores. Por fim serão apresentadas as implementações efectuadas na aplicação Android, *MyCar*, que fornece todas as funcionalidades previamente estabelecidas.

No capítulo 5 serão apresentados os testes efectuados e uma discussão sobre os resultados dos mesmos.

No capítulo 6 serão apresentadas observações e conclusões sobre o sistema e sobre os resultados dos testes anteriores. Será também apresentado o trabalho a realizar em futuros desenvolvimentos.

2. Estado de Arte

Este capítulo apresenta de uma forma geral os conceitos relevantes para o trabalho desenvolvido ao longo da dissertação.

Os tópicos seguem a mesma ordem que o trabalho tomou quando foi desenvolvido.

O capítulo 2.1 começa por apresentar, de um modo geral, o que são as redes móveis veiculares, explicando como funcionam e como podem ser implementadas e usadas, fornecendo assim uma contextualização ao leitor.

Os capítulos 2.2, 2.3 e 2.4 apresentam alguns sistemas possíveis de serem criados que façam uso das comunicações entre veículos, assim como a extensão para inclusão de dados do próprio veículo, criando assim um sistema complexo com várias tecnologias combinadas para criar uma aplicação final que forneça ao utilizador um bom conjunto de funcionalidades que o ajudam na condução.

2.1 Redes Veiculares

As redes veiculares são redes em que os nós são, como o nome indica, veículos. Estes nós criam uma rede do tipo *ad-hoc* [10], ou seja, criam uma VANET (Vehicular ad-hoc network).

Uma MANET (Mobile ad-hoc network) [11] é uma infra-estrutura de rede auto configurada, onde os nós são móveis e estão ligados via *wireless*. Os nós dentro desta rede são livres de se moverem em qualquer direcção, e são autónomos para criar novas ligações a outros nós de modo a manterem a conectividade na rede.

Assim uma VANET é uma subclasse da MANET, em que os nós móveis são veículos. Cada veículo funciona como nó ou *router*, na medida em que podem ser usados como retransmissores de uma mensagem, criando assim redes de maior alcance para ultrapassar o problema das ligações ponto-a-ponto entre os nós alcançarem apenas os 100-300 metros.

Além das comunicações ponto-a-ponto entre carros (V2V – Vehicle to Vehicle *communications*), é possível também instalar estações fixas nas estradas (RSU – *Road Side Unit*) que permitam aos veículos ligar-se a elas (V2I – Vehicule to Infrastructure *communications*) e fazer uso de um ponto de acesso à Internet, o que permite construir sistemas que utilizem servidores para base de dados, autenticações, autorizações, etc, como está visível na figura 1. O sistema não fica por isso limitado ao poder de processamento local que se possa colocar nos nós.

Para transformar o carro num nó duma rede veicular é preciso tecnologia e *hardware* próprio. É necessário que exista no veículo unidades *on-board* (OBU) que são responsáveis pela recepção e emissão das mensagens para a rede. Qualquer tipo de veículo pode conter uma OBU e deste modo ligar-se a uma rede veicular.

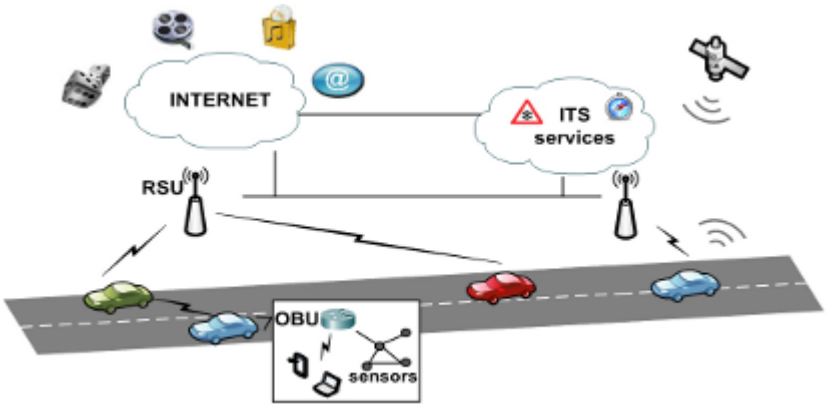


Figura 1 Imagem ilustrativa da VANET [34].

As redes veiculares tiveram um grande impulso quando acordos entre fabricantes e investigadores levaram à criação de novos projectos nesta área. Nos Estados Unidos foi reservada uma frequência para comunicações V2V e V2I, conhecido como *Dedicated Short Range Communication* (DSRC). DSRC é dedicado a comunicações de curta distância mas a grandes velocidades. Trabalha a uma frequência de 5.9 GHz e tem um alcance de cerca de 1000m. Em Julho de 2010, IEEE e ASTM adoptaram o DSRC (802.11p) como *standard*.

O desenvolvimento do DSRC levou ao aparecimento de projectos nos Estados Unidos como *Vehicle Safety Communications Consortium* [12]; no Japão houve um projecto semelhante foi criado com o nome de Internet Systems Consortium [13], e na Europa surgiu o projecto PReVENT [14].

O IEEE lançou o IEEE 802.11p como *ammendement* para que o *standard* de rede *wireless* locais (WLAN) suportasse o acesso *wireless* em ambientes veiculares (WAVE).

Para fazer face às especificações especiais das VANETs, o IEEE desenvolveu o protocolo WAVE composto pelo *standard* 802.11p (WAVE *standard*) juntamente com a família IEEE 1609 que compõe os *standards* que pretendem operar com IEEE 802.11p. Este tipo de *standard* é o que actualmente se utiliza nas ligações entre os nós e estrutura-se como ilustra a figura 2.

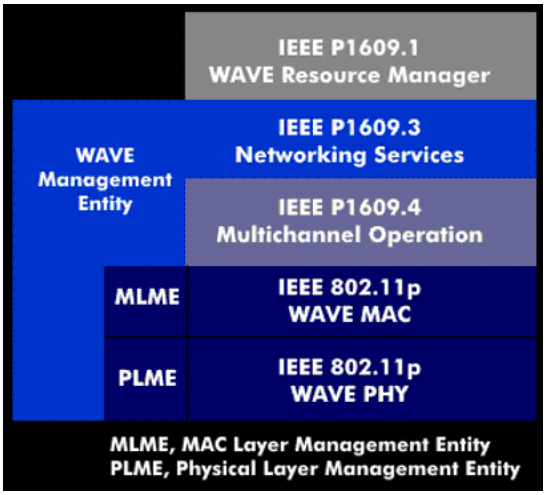


Figura 2 Protocolo 802.11 p e família 1609, ref [15].

A visão genérica da arquitectura de um sistema de comunicação entre veículos e infra-estrutura para criação de serviços que apoiem um Sistema de Transportes Inteligente pode ser observada na figura 3. No sistema criado neste trabalho, a comunicação entre os veículos não é necessariamente feita através das placas exteriores fixas, caso estes se encontrem dentro do alcance das antenas de 802.11/p existentes em cada veículo.

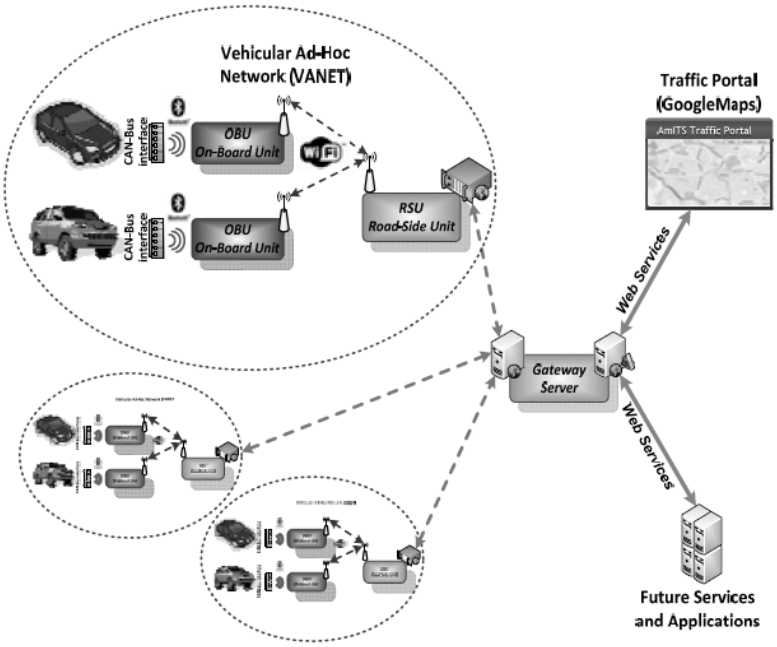


Figura 3 Componentes principais na arquitectura de um Sistema de Transporte Inteligente terrestre clássico [21]

As OBUs têm como principal papel reunir todos os dados possíveis de receber do veículo através da ligação à chamada ao canal de dados do veículo (OBD), que permite controlar e receber informação de dispositivos electrónicos do veículo, assim como grande parte dos novos sistemas de segurança dos veículos, e como tal fornece acesso a dados como velocidade, consumo, temperatura de diferentes partes do motor e rotações entre outras. Outros dados que as OBUs fornecem e são armazenados em base de dados remotas, provêm de equipamentos da própria placa, como por exemplo posição GPS, velocidade de circulação, matrícula (inserida manualmente aquando da instalação), identificador da OBU.

No seguimento deste trabalho, mas não fazendo parte do mesmo, foi criada uma interface que permite o acesso aos dados que for possível obter, passando-os à OBU que por sua vez, através da RSU, pode armazenar num servidor remoto. Este é o trajecto genérico dos dados para que se possam armazenar e tratar em grande volume. Contudo, não é o único percurso que os dados podem seguir como se irá abordar mais à frente.

Além do trabalho de envio dos dados obtidos do veículo, é à OBU que se ligam os dispositivos que os ocupantes, condutor e restantes, podem utilizar para integrar no sistema. Estes dispositivos podem

apenas servir para usufruir do acesso à Internet que a OBU oferece, ou podem ser dispositivos com aplicações semelhantes à aplicação que foi criada que visam oferecer serviços de segurança.

As RSUs têm a função de receber e enviar todos os dados que as OBUs têm para enviar e/ou receber. Estes equipamentos, situados ao longo das vias, estão ligados a um servidor que lhes fornece ligação a toda a Internet, e deste modo também as OBUs ficam com esse acesso à rede global. As RSUs podem também receber dados de outros elementos externos da via, como semáforos que tenham interfaces para ligar às RSUs, de modo a transmitir dados às OBUs no alcance.

Os dados podem ser usados para criar as mais diversas aplicações e serviços para os condutores ou ocupantes dos veículos. É sobre esta capacidade de comunicação com serviços remotos que assentam grande parte das aplicações que podem ser geradas nas diferentes áreas, desde segurança ao controlo de tráfego e ao entretenimento.

Neste trabalho o foco principal foi o uso das comunicações entre veículos para transferência de mensagens que possam enviar alertas, informações de outros veículos como posição e velocidade, de modo a poder usar a rede veicular e comunicação directa entre veículos mesmo quando o acesso às RSUs ou Internet não está disponível, fazendo também baixar os tempos de latência impostos pelas comunicações ao servidor e consequente tempo de resposta. Assim, espera-se atingir maior rapidez na partilha de mensagens de alerta importantes como acidentes ou até travagens bruscas do veículo da frente.

2.2 Redes veiculares e ITS

O objectivo principal das redes veiculares é criar uma rede de comunicação entre todos os veículos de modo a construir um sistema de transportes inteligente [3], que aumente a segurança nas vias de circulação [16]**.** ITS inclui todo o tipo de comunicações veículo-veículo (V2V), veículo-infra-estrutura (V2I) e dentro de cada veículo. Contudo, ITS não se restringe apenas a veículos, mas inclui também comunicações entre comboios, aviões e barcos.

Apesar das redes não estarem ainda completamente implementadas é possível pensar já em diferentes áreas para aplicações dentro do contexto de redes veiculares. A classificação das aplicações não é única, e cada instituição que desenvolve sistemas para transporte inteligente possuiu um conjunto de aplicações próprias diferente das outras empresas.

Contudo, genericamente distinguem-se 3 tipos de aplicações [3] [5] [17] [18]:

- Segurança Activa;
- Gestão do fluxo de trafego e melhoramento da condução;
- Entretenimento;

O sector da **Segurança Activa** é o objectivo primário das comunicações veiculares. Nesta categoria encontram-se várias aplicações que envolvem um veículo reportar aos outros anomalias que encontra na faixa de rodagem que possam colocar os outros utilizadores da mesma via em perigo.

Estes casos incluem avisar os outros veículos se o veículo em questão encontrar um obstáculo, reportar uma avaria de um outro veículo, uma paragem de emergência dele ou de outro veículo, piso em más condições, trânsito congestionado, condições atmosféricas perigosas, acidente etc.

O modo como é efectuado o reconhecimento do perigo por parte de cada veículo pode ser feito através de sensores que detectem o perigo, sejam obstáculos, outros veículos, travagem abruptas, mau piso, etc, ou através da intervenção humana, condutor ou ocupante que através de um *software* com interface apropriada permita o envio, de forma simples e rápida, do alerta para os outros veículos.

No sector da **Gestão do fluxo de trafego e melhoramento da condução,** podemos incluir várias aplicações diferentes como sistemas de informação do tráfego, informações sobre o clima presente/ futuro, estações de serviço e restaurantes com informação de preços. Todas estas funcionalidades podem transmitir informações aos condutores de modo a planear com antecedência todo o trajecto da viagem que pretendem fazer fazendo deste modo com mais segurança e evitando assim zonas de congestionamento. Informação sobre velocidade da via em circulação pode também ser mostrada ao condutor, sendo um factor de segurança. A aplicação deve fornecer ao condutor os melhores itinerários para a rota pretendida, com base em dados como trânsito das vias, estado do piso, estado do tempo, acidentes relatados, velocidades permitidas para circulação. Deve também ser capaz de recalcular um novo itinerário caso o condutor indique a impossibilidade de ir pelo caminho que foi aconselhado.

Por fim na secção de **Entretenimento** inclui-se todas as funcionalidades que possam servir de distracção a todos os ocupantes do veículo, expecto idealmente, o condutor. Com acesso à Internet disponível, as aplicações de entretenimento são variadas como ouvir música, ver um filme, ler livros *on-line*, aceder às contas de *e-mail*, jogar online, procurar pontos de interesse perto.

Outra funcionalidade que pode ser inserida neste campo, uma vez que está mais virada para o acesso através da Internet, é o acesso a informação relativa ao veículo remotamente. A possibilidade de aceder em qualquer lado a detalhes do veículo como posições GPS, que pode ser útil se o ambiente for de controlo de uma frota onde interessa a visualização de rotas, velocidades de circulação dos veículos, aceder a um histórico de mensagens recebidas pelo carro, seja de "chat" com outros veículos ou alertas enviados/recebidos pelo veículo, assim como dados de consumos efectuados.

Estas aplicações devem ser criadas de modo a mostrar a informação possível do modo mais simplista e prático possível, para que os utilizadores, como o condutor, não se distraiam na sua utilização. As soluções passam muitas vezes por recorrer a mapas com indicadores, uso de mensagens de texto para alertas simples, assim como ecrãs que sejam de fácil visualização e com o mínimo de interacção necessária da parte do condutor para que a aplicação funcione. Apesar de ser reconhecida a importância deste aspecto na utilização da aplicação, não foi tomado como prioridade durante o processo de desenvolvimento da aplicação. Deste modo num trabalho futuro fará sentido melhorar certos aspectos da interface.

2.3 ITS e Projectos Relacionados

Ao longo dos anos têm sido criados alguns projectos que visavam criar uma rede com os veículos que permitisse troca de mensagens entre eles e com o exterior. Exemplos disso, são estudos como o "*Drive-thru Internet"* [19] feito por uma universidade de Helsínquia, que tentava implementar e testar a viabilidade de uma rede deste género mas usando apenas protocolos ditos "normais" de redes Wi-Fi (802.11b e 802.11g). A ideia era criar zonas de ligação (*HotSpot*) ao longo das estradas em que os veículos se ligavam por Wi-Fi a esses *HotSpots* e durante um curto período de tempo, consoante a velocidade do mesmo e o alcance da antena Wi-Fi, os condutores teriam acesso á Internet. Criando uma rede *ad-hoc* entre os veículos foi possível adicionar a possibilidade dos condutores comunicarem entre si mesmo quando fora de zonas com acesso a Internet.

Um estudo para verificar a viabilidade da utilização destas normas (802.11b e 802.11g) em cenários de redes veiculares levado a cabo na universidade de Stanford [20] mostra que a taxa de transmissão de dados tem valores significativamente altos, apesar que tem de se ter em conta que o teste foi feito com pacotes UDP, em *broadcast*, o que aumenta a taxa de transmissão, contudo diminuiu bastante conforme a distância aumenta.

Após surgir uma novo protocolo, dedicado a redes veiculares (802.11p) surgiram novos projectos.

Nos Estados Unidos surgiu o projecto *Vehicle Safety Communications Consortium* [12], que objetivava juntar investigadores de telecomunicações e algumas construtoras de automóveis como Nissan, Toyota, Ford, BMW, para com a ajuda do DSRC delinear as principais aplicações possíveis de construir que viessem ajudar na condução e/ou prevenção de acidentes.

Num trabalho realizado por Unai Hernandez [21], foi desenhado um protótipo de OBU que envolve todas as interfaces necessárias para criar as comunicações necessárias para existir acesso a informação do interior do veículo através de ligações ao bloco central de "controlo" do veículo, de estruturas externas fixas, de outros veículos equipados com as mesmas OBUs ou do utilizador através de uma interface visual. O sistema criado e testado é para ser incorporado em cada veículo que em conjunto com estações externas fixas, RSU´s permite criar um sistema de grande escalabilidade que sirva de base a serviços que ajudam a montar um Sistema de Transporte Inteligente. Além deste protótipo de OBU foram criados dois serviços que visam mostrar que este sistema criado é útil e deve realmente ser usado. Estes serviços visam duas áreas muito importantes no âmbito de ITS, condução ecológica (eco-driving*)* e relatórios de tráfego (*traffic reports)* que demonstram no fim do artigo terem atingido valores de importante em poupança do ambiente e de combustível. Foi um sistema criado em arquitectura *bottom-up*, pelo que o sistema ficou o mais genérico possível podendo ser usado para criar mais aplicações/serviços conforme surgir a necessidade.

Recentemente, Fevereiro de 2014, foi anunciado que o Departamento de Transportes dos Estados Unidos da América iria avançar com projectos que visam criar ligações *vehicle-to-vehicle (V2V)* para

veículos ligeiros [22]. Estes projectos têm como objectivo principal explorar uma das principais aplicações das VANETs, aumentar a segurança nas estradas permitindo que os veículos comuniquem uns com os outos tentando reduzir o número de colisões através da troca de mensagens sobre posição e velocidade dos veículos.

Esta vontade do Departamento de Transportes não é nova. Já em 2011 [23] [24] tinham mostrado interesse em criar este tipo de ligações, para tentar reduzir os drásticos números de vítimas de acidentes rodoviários (cerca de 33 mil mortes em 2009) assim como os gastos envolvidos em operações relacionadas com os acidentes e em perdas de produtividade (75 mil milhões de dólares americanos).

2.4 Redes Veiculares Aplicações Móveis

Relativamente a criação de aplicações que interajam com as redes veiculares existem alguns casos de sucesso. Um exemplo foi o trabalho desenvolvido por M. Esbjörnsson, que se traduziu numa aplicação móvel, com vista a melhorar os meios de comunicação para condutores de motas [25]. Tendo como base as dificuldades relatadas por grupos de motociclistas em comunicar e organizar os grandes grupos que costumam circular na estrada, foi criada uma aplicação móvel baseada numa rede Ad-Hoc em que os nós da rede são as motas.

Gianpaolo Macario [26] descreveu como projectar Software de informação e entretenimento de âmbito veicular, para utilizar nos *smartphones* com sistema operativo Google Android. Um dos principais problemas dos construtores deste tipo de Software designado "Infotainment", que se traduz em aplicações que juntem visualização e tratamento de informação e uma componente de entretenimento que visa atrair os utilizadores, é a grande variedade de veículos que existe. Não é viável construir uma plataforma dedicada para cada marca e modelo. De modo a reduzir custos e possibilitar um melhor e mais rápido desenvolvimento de aplicações comuns a vários veículos, o autor propõe a criação de uma camada de abstracção que corre num sistema comum a vários dispositivos, o sistema operativo Android do Google. Contudo neste trabalho desenvolvido por Macario não existe uma placa OBU onde o *smartphone* se liga. A biblioteca visa ligar directamente as aplicações Android com os dados de OBD do veículo de modo a puder recebe-los e trata-los.

Recentemente foi também lançada pela Apple uma plataforma, *CarPlay* [27], que permite a interacção entre o veículo e os *smartphones* e *tablets* da marca, respectivamente *iPhone* e *iPad*.

Esta aplicação permite que o utilizador através do ecrã do seu veículo possa aceder aos conteúdos do seu telemóvel ou *tablet* e fazer uso de aplicações de uso comum como efectuar e receber chamadas ou mensagens, aceder à música do utilizador, dados na *cloud* entre outras funcionalidades. Esta tentativa de ligar aparelhos pessoais com o veículo pode ser visto como uma demonstração de uma tendência que existe de aproximação entre construtoras de automóveis e empresas ligadas ao ramo do

mobile, muito anunciada alguns meios de comunicação que seguem as novidades tecnológicas [27] [24] [28].

Outro exemplo de aplicação é o *DriverAsist* [33] desenvolvido por Diewald. Neste trabalho foi criada uma aplicação desenvolvida em Android que mostra informação de trânsito proveniente da rede veicular e de uma central de dados de trânsito. Utiliza comunicação V2V e V2I para comunicar e transferir informação sobre o estado do trânsito em redor do utilizador. Este sistema suporta ainda um sistema de alertas sobre acidentes.

2.5 Sumário

Após a pesquisa efectuada verifica-se que existe muito trabalho efectuado no que respeita a tentativas de criar ambientes que propiciem o aparecimento de sistemas para melhorar as condições de ajuda para quem circula nas estradas. Os vários projectos mundiais referenciados são prova disso.

Contudo, apesar de ser reconhecido como uma solução a explorar devido às elevadas capacidades, a integração com *smartphones* ainda não está disponível em pleno. Os exemplos demonstram que a principal falha está na dificuldade que existe em conectar à mesma rede onde os carros comunicam, os telefones inteligentes dos utilizadores/ condutores.

Nos trabalhos já efectuados em que esta falha foi colmatada também não foram disponibilizadas ao utilizador todas as funcionalidades que poderiam ser aproveitadas, além do sistema de alerta de tráfego ou da troca de informações com servidores remotos para localizações ou apenas divertimento.

Propõe-se então a criação de um sistema abrangente, que integre de forma mais completa o *smartphone* com a rede veicular, com o veículo e com a Internet de modo a fornecer uma vasta lista de funcionalidades. Desde alertas de mapas, alertas do veículo, comunicação entre utilizadores por texto, acesso a dados do veículo, criação e visualização de rotas e acesso a dados remotos e serviços *web*.

Este sistema prevê colmatar a falha de acesso à rede veicular disponibilizando uma camada de acesso utilizável por terceiros para o desenvolvimento das suas aplicações.

3. STRIVE: Using Smartphone to share TRaffic Information in VanEt

Neste capítulo vamos apresentar o sistema STRIVE- Partilha de informação de trânsito citadino com *smartphones*. Vão ser apresentados os vários componentes e API do STRIVE que permitem obter uma aplicação para auxiliar a condução em questões de consumo, gestão de viagens e percursos de curta ou longa distancia, manutenção de veículos, sistemas de alertas em tempo real sobre possíveis perigos entre outras aplicações.

No processo, abordamos em mais detalhe o REINVENT [9], que foi a plataforma que serviu de base a este trabalho, assim como as extensões que foram necessárias para dar suporte à aplicação final.

A nível de funcionalidades, o sistema deve satisfazer necessidades do utilizador relacionadas com a VANET - como por exemplo, obter e enviar alertas sobre condições da estrada, do trânsito, ou falhas mecânicas dos veículos, que pudessem por em perigo os outros condutores, de modo a que os mesmos pudessem tomar decisões para evitar filas ou mesmo acidentes; enviar e receber mensagens de texto que permitisse aos ocupantes manter conversas enquanto seguem nos respectivos veículos, e ainda ser abrangente o máximo possível para suportar novos tipos de mensagens na rede veicular.

A nível de ligação com o veículo, o condutor deveria ser capaz de poder observar em tempo real dados relacionados com o veículo, como temperatura, consumos do motor, problemas detectados. O sistema deve também oferecer a possibilidade de mostrar ao condutor um histórico das viagens efectuadas, mostrando num mapa as deslocações feitas assim como as médias de consumo e velocidade para as várias partes do percurso em observação.

O condutor deve ter a possibilidade de visualizar e analisar estes dados em qualquer lugar através do acesso a uma base de dados por meio de um *website* que contenha todos os dados do veículo assim como das rotas efectuadas.

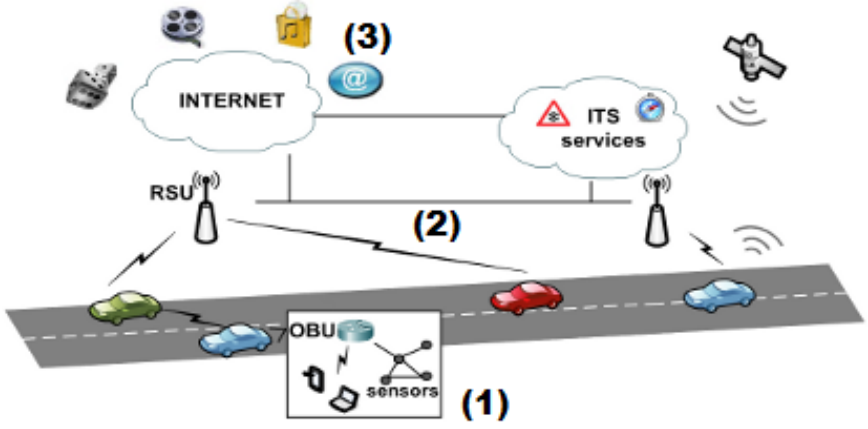


Figura 4 Imagem ilustrativa da VANET. Ref [29] com indicações para ilustrar os 3 objectivos principais do trabalho. Número (1) – ligação ao sensores do veículo, Número (2) – assegurar conectividade entre veículos, Número (3) – disponibilizar acesso a recursos remotos

Para garantir os objectivos propostos é necessário garantir que o STRIVE permita:

- Ter os valores dos sensores disponíveis na aplicação, ilustrado no número (1) da figura 4: é necessário que exista uma ligação da OBU à OBD do veículo, já disponível nas OBDs utilizadas, que não foi desenvolvido no âmbito deste trabalho. Contudo é necessário fazer a leitura desses valores na OBU e fazê-los chegar à aplicação.
- Ter comunicação entre veículos das mensagens da aplicação, ilustrado no número (2) da figura 4: é necessário que a camada de abstracção esteja totalmente funcional para envio e recepção de mensagens. Para criar esta camada de abstracção é necessário implementar código quer nas OBUs quer no *smartphone*.
- Fazer uso de recursos externos, ilustrado no número (2) da figura 4, como por exemplo da base de dados (BD) comum ao serviço: é necessário criar os *web-services* para operações sobre a BD. Considerando que existe conectividade com RSU e acesso à Internet, a aplicação deve ser capaz de enviar os pedidos de forma simples e obter as respostas pedidas aos serviços.

3.2 Funcionalidades principais do sistema

O STRIVE i.e. o sistema incluindo a aplicação e servidores remotos, deve assegurar as seguintes funcionalidades:

- Manter um repositório central com a informação relativa a GPS, velocidade e consumo de cada veículo equipado com OBU. Essa informação, após passar por filtros relacionados com questões de direito de acesso, pode ser mostrada aos utilizadores através da aplicação ou através de um *web-site,* o que se aplica melhor num contexto de gestão de frotas de uma empresa;
- Receber e difundir mensagens de e para os veículos nomeadamente de informações sobre problemas relacionados com estradas, trabalhos na via, trânsito, sinalização e operações policiais adaptando-se às condições de conectividade, seja via acesso à Internet disponível ou da VANET para passar a informação da aplicação para outros veículos (aplicações) e/ou repositório central.

Para alcançar este objectivo, o STRIVE deve suportar/facilitar aplicações com a ligação Wi-fi 802.11 /b/g/n, dentro do veículo equipado com uma OBU.

Foram também identificados alguns requisitos mais especificos da aplicação. Esta deve manter um histórico de rotas do veículo dos últimos 5 dias, podendo carregar mais dados remotamente se houver um acesso à Internet. Para cada rota, a aplicação deve mostrar ao condutor o máximo de informação do veículo disponível (consumos instantâneos, consumos médios, distâncias percorridas), para cada ponto GPS. Deve também fornecer um feedback visual sobre quais as zonas de maior e menor

consumo médio, para que o utilizador no futuro possa alterar a sua rota a fim de evitar consumos elevados.

A aplicação também deve permitir comunicar com os veículos em redor através de mensagens de texto escritas, pelo ocupante, e manter um histórico das conversas que foram efectuadas com cada veículo.

3.3 REINVENT

REINVENT [9] é um sistema que cria uma abstracção da camada de transporte sobre redes veiculares, que permite às aplicações móveis o envio de mensagens, com todos os tipos de informação, através, usando uma interface de acesso normalizada baseada em REST. A implementação existente é implementada no sistema operativo Android para receber e enviar informação entre OBU e aplicação.

No entanto, na sua versão actual, o REINVENT apenas possui suporte para um tipo de mensagens-texto- entre veículos, e não permite endereçar as mensagens utilizando o endereço específico do veículo na VANET.

3.3.1 Arquitectura do REINVENT

O REINVENT é um sistema que envolve o uso de outras estruturas e está implementado em mais do que uma plataforma, e para melhor entender o mesmo é necessário fazer uma diferenciação dos diferentes módulos e as funções específicas de cada um.

Na figura 5 pode observar-se o módulo de REINVENT instalado no *smartphone* e o módulo da OBU cujos detalhes serão abordados posteriormente. Na figura pode-se observar que o módulo que representa a aplicação (*MyCar*) no *smartphone*, faz uso de vários serviços. Esses serviços garantem uma ligação por tipo de mensagem ao REINVENT onde são feitas as "escutas" e recepção das várias mensagens. No módulo REINVENT instalado também no *smartphone* é visivel o módulo interno de RabbiMQ para gestor de mensagens e encontra-se ligado aos serviços por ferramentas internas do sistema Android. Na figura está ilustrado também o módulo do REINVENT instalado na OBU com as diferentes interfaces disponíveis para enviar/receber dados.

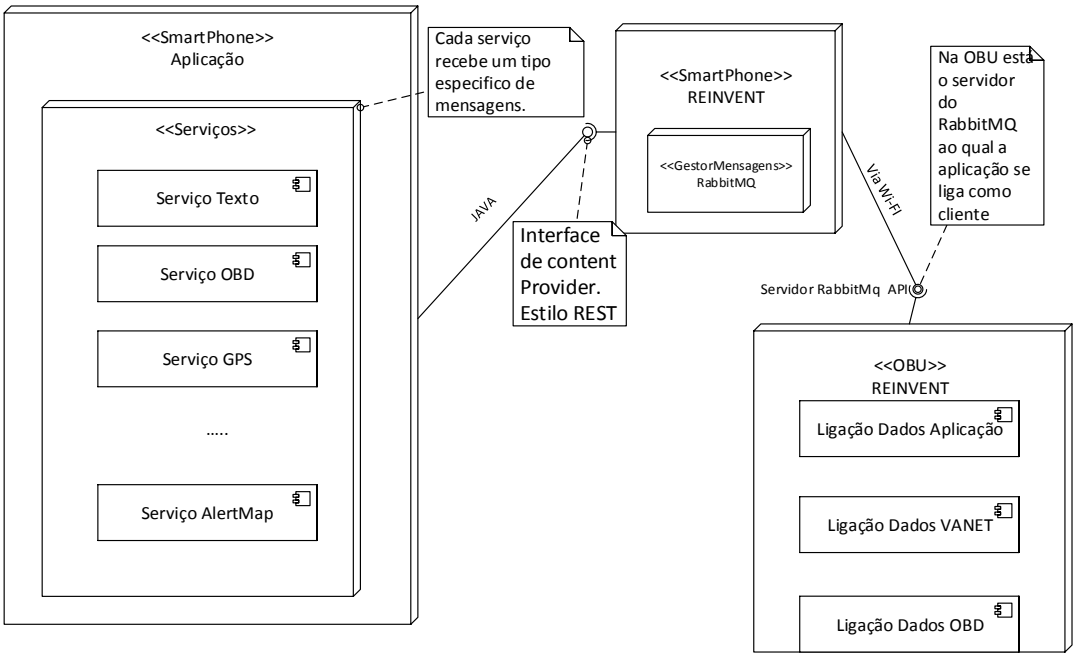


Figura 5 Diagrama de Componentes - Módulos do REINVENT

A comunicação entre módulos que compõem o REINVENT segue o paradigma de troca de mensagens como vai ser abordado mais adiante no documento.

Este sistema foi criado no âmbito de um trabalho anterior; contudo, neste trabalho esse sistema inicial foi alterado e melhorado para responder às novas necessidades que as aplicações podem exigir, e sem entrar em detalhes técnicos irá ser explicado o funcionamento do sistema REINVENT e como qualquer aplicação Android pode utilizá-lo para se ligar à rede veicular e/ou a Internet.

A preocupação principal foi criar um sistema que permitisse a uma aplicação enviar e receber mensagens para outros veículos ou mesmo para servidores remotos de modo abstracto para a mesma, apenas enviando e recebendo mensagens usando a ligação Wi-Fi 801.11 n/g suportada pelo sistema Android.

3.3.2 REINVENT e serviços REST

A arquitectura do sistema é baseada no estilo de arquitectura REST (Representational State Transfer). REST é um modelo Cliente - Servidor, bastante usado em HTTP para navegar de uma página web para outra, através de envio de mensagens com pedidos que permitem avançar na navegação sem obrigar a aplicação ou o servidor a guardar dados sobre o estado da ligação como por exemplo os conhecidos *cookies*.

No modelo REST os recursos que pretendem ser acedidos são identificados de forma única através de um identificador único, *URI (Uniform Resource Identifier)*. Da mesma forma foi adaptado para o REINVENT este sistema de identificação de forma única de recursos.

O autor Roy Fielding, considerado o "pai" da arquitectura REST, que em 2000 escreveu pela primeira vez o termo na sua tese de Doutoramento [35], e enumerou as principais características, das quais se destacam para este trabalho duas destas características.

A principal característica, já enumerada, é a Cliente – Servidor. Esta característica tem por base a separação dos conceitos, cliente e servidor. Separando a interface, (Cliente) do armazenamento e processamento de dados (Servidor) garante-se uma melhor portabilidade da interface por múltiplas plataformas e melhora a escalabilidade simplificando o componente do servidor. O funcionamento desta arquitectura garante que vários clientes possam estar ligados ao servidor ao mesmo tempo fazendo pedidos, que são processados e cujas respostas são interpretadas e mostradas na interface ao utilizador.

A segunda característica é a comunicação *Stateless*, ou seja, cada pedido que o cliente envia ao servidor deve conter na mensagem informação necessária para o servidor compreender o pedido e poder enviar uma resposta correcta. O estado da sessão é mantido do lado do cliente. Esta característica aumenta propriedades como visibilidade, confiança e escalabilidade. A visibilidade aumenta porque um sistema de monotorização consegue facilmente observando a mensagem do cliente saber todos os parâmetros do pedido. A confiança é aumentada porque se torna mais simples a tarefa de recuperar a partir de falhas parciais. A escalabilidade aumenta pois o servidor como não tem de guardar estados das comunicações com os diversos clientes pode libertar recursos mais rapidamente, e torna mais simples a implementação dos servidores e/ou de novos módulos.

No âmbito do projecto REINVENT foi adaptada a arquitectura REST para criar uma camada de abstracção entre a camada de transporte e a camada de aplicação. O funcionamento do REINVENT permite assim, que as aplicações (clientes) enviem pedidos a um módulo Android do REINVENT que funciona como servidor para aceder a recursos de forma invisível para a aplicação através de URIs registados no servidor. Estes recursos não são mais do que serviços, como envio de mensagens para a rede, recepção de mensagens, ou obtenção de endereços de outras entidades na rede através do módulo de DNS que está a ser desenvolvido este ano num terceiro projecto, que garantem a comunicação entre as aplicações e as OBUs, acendendo desta forma quer a rede veicular quer a Internet, se disponível.

3.3.3 REINVENT e RabbitMQ

Software para gerir o envio e recepção de mensagens através da criação e gestão de filas de mensagens.

Segue uma arquitectura cliente – servidor, como mostra a figura 6.

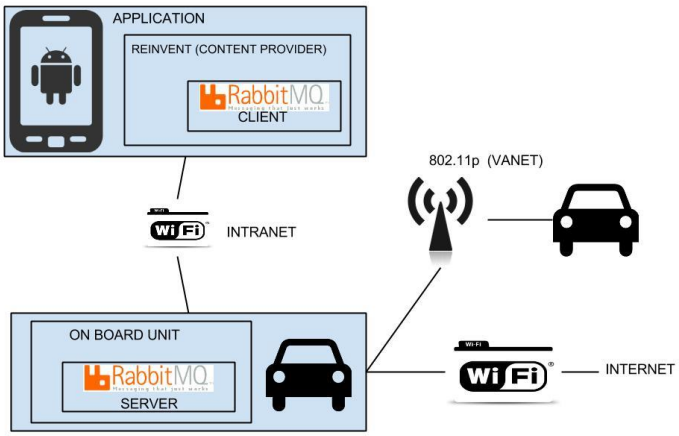


Figura 6 Organização do RabbitMQ nos módulos do REINVENT

Neste caso o módulo do REINVENT é o cliente que se conecta ao RabbitMQ [30] servidor que está instalado na OBU. O cliente quando pretende enviar uma mensagem indica o nome da fila de mensagens em que a mensagem vai ser colocada. Essa fila se ainda não existir é criada no servidor e a mensagem é adicionada. A própria OBU é também cliente quando pretende enviar uma mensagem para a aplicação.

A notificação por parte do servidor aos clientes que existem novas mensagens na fila e consequente envio para os mesmos é garantido de forma interna pela biblioteca do RabbitMQ-Cliente, ficando transparente até para quem programa e cria o sistema de envio e recepção de mensagens.

3.3.3.1 Troca de mensagens no REINVENT

Todo o processo de comunicação entre aplicações e OBU foi pensado de modo a seguir este modelo. Como se pode ver na figura 6, as aplicações desenvolvidas irão aceder a um módulo do REINVENT como cliente, que é no fundo uma aplicação Android de apoio para outras posteriores, e que deverá ser pré-instalado no *smartphone*, quando pretendem um determinado serviço. Esse módulo transformará os pedidos das aplicações em mensagens específicas e comunicará através do envio dessas mensagens para um módulo servidor instalado nas OBUs dos veículos, onde existe também um gestor de filas de mensagens. No caso de o servidor ter mensagens para entregar às aplicações, por exemplo respostas aos pedidos feitos, irá acontecer o inverso. O servidor enviará para o gestor de filas de mensagens as respostas e o mesmo informará o módulo cliente de novas mensagens. Aquando da recepção as mensagens recebidas são tratadas e a aplicação será informada da sua recepção.

Como se pode observar todo o sistema é assente na comunicação dos vários elementos, desde a aplicação que o utilizador possui, até ao servidor mais remoto e consequente resposta.

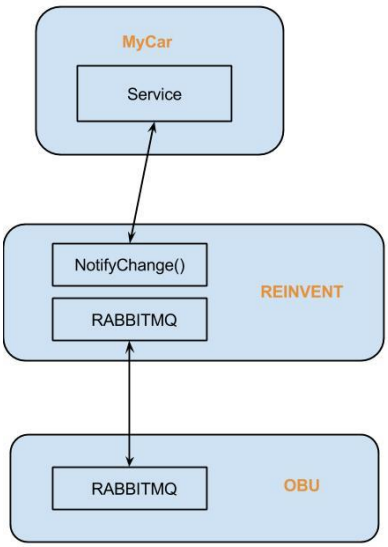


Figura 7 Módulos de RabbitMQ nos componentes da Aplicação, REINVENT e OBU.

Uma das definições mais usadas quando se trata de modelos baseados em troca de mensagens é a que foi dada por Curry, onde diz que estes modelos (MOM – Message Oriented Middleware), são *software* e/ou *hardware* desenhados para fornecer métodos de comunicação entre entidades heterogéneas, o que acontece neste projecto ao tentar conectar o cliente com a rede veicular.

Recorrendo a estruturas de UML é possível obter diagramas de mensagens representativos da sequência de mensagens possíveis de trocar nestes modelo MOM quando queremos realizar uma determinada tarefa. Na figura 8 podemos observar o caso genérico de envio de mensagem de texto de uma aplicação para outra, como o caso de uma conversação entre utilizadores, em que fica completamente escondido e abstraído o processo de comunicação para os utilizadores.

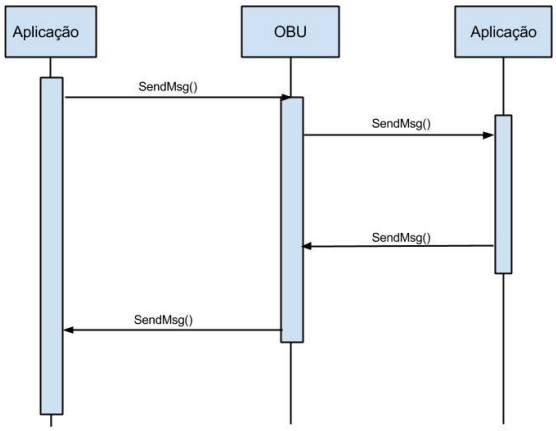


Figura 8 Diagrama de troca de mensagens entre aplicações

3.3.4 Módulo REINVENT no *Smartphone*

Este módulo é uma aplicação Android independente, que deve ser instalada antes de qualquer aplicação que queira ligar-se à OBU. É responsável pela criação e registo dos URIs que serão depois utilizados pela aplicação, para efectuar diversas aplicações.

Foi criado inicialmente no projecto do aluno Filipe Oliveira no ano passado e foi já no de correr deste projecto modificado e ampliado.

Para implementar uma arquitectura do estilo REST, entre este módulo, servidor, e as aplicações, clientes, foi usado um componente do sistema Android chamado *Content Provider* que será abordado em detalhe no capítulo 4. Resumidamente, um *Content Provider* permite a criação de uma camada de abstracção (também conhecida como interface) entre uma fonte de dados, sejam base de dados, ficheiros, ou acesso remoto, e uma aplicação. Na sua implementação são criados URIs através do url de Autoridade (Authority), junto com o url de caminho (Path) único para cada recurso.

Esse URI tem na primeira parte, o endereço do Content Provider que está a ser usado, e na segunda parte uma indicação de quais os dados que sofreram alteração. Esses dados são tratados como se de "tabelas" se tratassem e são identificados pelo *path* (URI do Content Provider + URI especifico de cada tabela), de modo que essa notificação funciona como um aviso de uma alteração á "tabela" dada pelo *path* passado. O mesmo principio se aplica quando se pretende criar uma escuta de notificações de alterações feitas.

É através destes URIs que as aplicações comunicam com o Content Provider, e este com as aplicações. Um exemplo simples é a recepção de mensagens da OBU pelo Content Provider e passagem à aplicação: o REINVENT recebe a nova mensagem da OBU via *wireless* através do cliente de gestor de mensagens. Verifica o tipo de mensagens e lança uma notificação de alteração ao URI correspondente ao serviço, por exemplo **"com.example.vanetprovider.networkprovider/newTXTMsg"**. Do lado da aplicação existe um serviço que se encarrega da "escuta" das alterações ao URI e de pedir ao REINVENT essa nova mensagem, tratando-a depois.

Além da ligação com as aplicações de alto nível, este módulo é também responsável pela conexão a placa do veículo. Para tal faz uso de uma biblioteca externa para cliente de um gestor de filas de mensagens, o RabbitMQ, criando assim um sistema MOM como foi falado previamente.

3.3.5 Módulo REINVENT na OBU

Este módulo tem 2 funções de grande importância. Por um lado é onde o servidor do RabbitMQ está instalado e a funcionar. É a este módulo que se devem ligar todos os clientes para que possam enviar/receber mensagens de e para a OBU. Por outro lado, é este módulo que fica encarregue de

colectar as mensagens que chegam a OBU, quer das RSU quer da interface com o veículo onde a OBU está instalado, e enviar toda essa informação para o módulo REINVENT que está a instalada no *smartphone*.

Usando ciclos de escuta contínuos, a OBU escutará ao mesmo tempo as várias interfaces. A ligação ao veículo por parte da OBU, não está no âmbito do projecto pelo que o acesso aos dados deve ser fornecido pela OBU.

Após receber novas mensagens o REINVENT está encarregue de faze-las seguir pela interface que cada mensagem deve ter como destino, seja enviar para a aplicação, para a rede veicular ou para a Internet.

3.3.6 Filas de Mensagens no RabbitMQ

Como foi explicado, o RabbitMQ é um gestor de mensagens usado para facilitar o processo de comunicação dos vários componentes.

Todo o processo de recepção e envio de mensagens se baseia na criação de filas específicas de mensagens que quer o servidor quer o cliente podem registar para ficar a escuta ou para inserir nelas novas mensagens.

No desenvolvimento do sistema foram criados 10 tipos diferentes de mensagens para assegurar todas as funcionalidades.

Contudo, foi necessário criar mais do que 10 filas de mensagens pois optou-se por usar cada fila apenas num sentido. Ou seja para o mesmo tipo de mensagem podem existir duas diferentes filas, como por exemplo para mensagens de texto, como se pode observar na tabela 2, onde se pode ver em cada módulo qual a fila que se usa para enviar cada tipo de mensagens.

No caso da recepção de mensagens, são criadas "escutas" nas filas com o nome usado para enviar no outro módulo.

Tipo de Mensagens	OBU	Android
Alertas de Mapa	CarToPhoneALERTMAP	PhoneToCarALERTMAP
Alertas de Veiculo	CarToPhoneALERTVEHICLE	PhoneToCarALERTVEHICLE
Texto	CarToPhoneTXT	PhoneToCarTXT
StatusRequest	-	PhoneToCarSTATUS
StatusResponse	CarToPhoneSTATUS	-
DnsRequest	CarToPhoneDNSREQUESTMSG	PhoneToCarDNSREQUESTMSG
DnsResponse	CarToPhoneDNSRESPONSEMSG	PhoneToCarDNSRESPONSEMSG
http	-	PhoneToCarHTTPMSG
GPS	CarToPhoneGPS	-
OBD	CarToPhoneOBD	-
HttpResponse	CarToPhoneHTTPRESPONSE	-
Custom	CarToPhoneCUSTOM	PhoneToCarCUSTOM

Tabela 1 Filas para envio em cada módulo dos vários tipos de mensagens

3.4 STRIVE e os cenários

Como foi abordado anteriormente, para tornar possíveis todas as funcionalidades da aplicação, é necessário implementar e resolver 3 problemas principais: comunicação na VANET, comunicação com o veículo e leitura de sensores, e comunicação com o exterior. Esses três problemas cujos cenários já foram apresentados previamente implicaram soluções e implementações próprias. Através da figura 4, é possível identificar visualmente os 3 cenários principais utilizando uma imagem representativa de uma possível rede veicular, com a numeração dos cenários (Número 1 – ligação ao sensores do veículo, Número 2 – assegurar conectividade entre veículos, Número 3 – disponibilizar acesso a recursos remotos.).

3.4.1 Comunicação na VANET

Para dar suporte à comunicação sob rede veicular é necessário que nos equipamentos presentes no veículo, OBU, se faça a conversão das mensagens recebidas da aplicação pelo módulo REIVENT para mensagens que possam ser enviadas via WAVE para outra placa de outro veículo ou para uma RSU.

Esse módulo de conversão deve funcionar em modo bidireccional, ou seja também é necessário que haja a conversão de mensagens recebidas pela antena de WAVE para mensagens que possam ser inseridas nas filas do RabbitMQ (REINVENT) para ser enviadas para a aplicação.

Existem alguns tipos de mensagens da aplicação que devem ser propagados na rede mesmo quando não existe um acesso a Internet. Ou seja, é importante manter os objectivos principais das VANETs

que são o envio de alertas entre veículos que melhorem as condições de trânsito e aumentem a segurança de condutores e ocupantes.

Veja-se o exemplo específico para o caso de envio de uma mensagem de texto entre dois ocupantes de veículos diferentes. Pode-se observar alguns dos requisitos básicos: a aplicação deve ter suporte para criar mensagens de vários tipos, neste caso de texto. A partir de uma actividade, deve haver uma "conversão" da mensagem para WAVE de modo autónomo, a rede deve ter suporte para o envio de mensagens deste tipo, deve haver uma conversão automática de WAVE para Wi-Fi, deve existir no telemóvel, uma "escuta" às novas mensagens, e uma actividade para mostrar a mensagem recebida no contexto correcto, neste caso uma janela de conversação, ver figura 9.

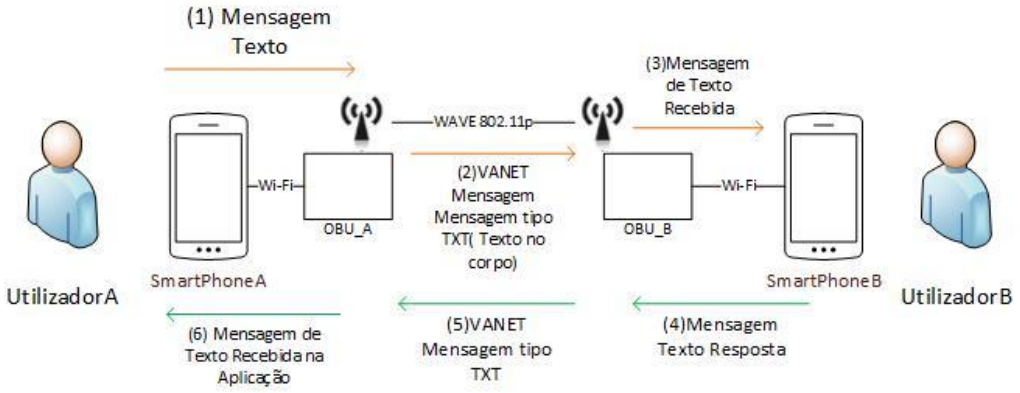


Figura 9 Representação de implementação do caso de uso para comunicação entre veículos.

A figura 10 representa o mesmo fluxo de informação, mas utilizando um diagrama de sequência com um exemplo de envio de mensagem para a rede veicular em que não é necessário resposta por parte do outro utilizador. Nesse diagrama são visíveis os vários componentes que fazem parte do processo de troca de mensagens de alerta, desde que o alerta é lançado por um utilizador no seu *smartphone* até que chega ao *smartphone* de outro utilizador na rede. É ainda visível o formato que cada mensagem ganha em cada comunicação entre elementos.

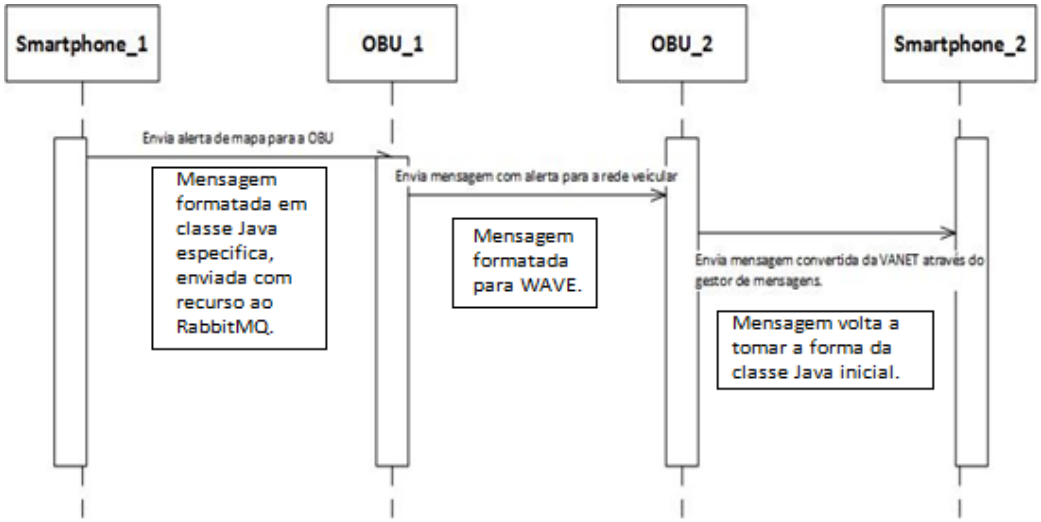


Figura 10 Diagrama de sequência para envio de alerta de mapa para a VANET

3.4.1.1 Formato das mensagens

O módulo de conversão está instalado na placa OBU e tem acesso às filas de mensagens do RabbitMQ.

Apesar de algumas funcionalidades da aplicação estarem dependentes do acesso à Internet, é necessário que pelo menos os alertas que a aplicação envia se propaguem para os veículos em redor.

Para tal, foi suportado a conversão de mensagens de alertas relacionados com mapa (problemas na estrada, sinalização, problemas de tráfego) e de alertas relacionados com o veículo que possam pôr em risco os outros automóveis como acidentes ou avarias na estrada.

Outra funcionalidade que pode ser mantida usando apenas rede veicular é a troca de mensagens de texto entre ocupantes dos veículos, sendo por isso necessário fazer essa conversão desse tipo de mensagens também.

Os tipos de mensagens enviadas pela aplicação através do RabbitMQ seguem os modelos, listados em pormenor no ANEXO B, seguintes:

- Mensagem de alertas relacionados com o mapa;
- Mensagem de alertas relacionados com o veículo;
- Mensagem de texto;
- Mensagem de StatusRequest;
- Mensagem de StatusResponse;
- Mensagem de DnsRequest;
- Mensagem de DnsResponse;
- Mensagem de Http;
- Mensagem de HttpResponse;
- Mensagem de GPS;
- Mensagem de OBD;
- Mensagem de Utilizador (Custom)

3.4.1.2 Conversores genéricos para a rede VANET

No caso do envio de mensagens da aplicação, ou da própria OBU, para a rede veicular é necessário criar um serviço de tradução dessas mensagens, recebidas pelo RabbitMQ da OBU e o módulo de comunicação para a VANET. Esse módulo de comunicação, WAVE não possuiu ferramentas que permitam a tradução automática das mensagens lidas do RabbitMQ para a norma 802.11 /p (WAVE). Assim tornou-se necessário a criação de módulos internos na OBU para efectuar essa conversão. Esses módulos foram chamados de "*Inside Translator"* (Conversor Interno) e "*Outside Translator"* (Conversor Externo):

- *Inside Translator* – Este módulo estará à "escuta" de novas mensagens na interface que liga à VANET e ficará responsável pela conversão dessas mensagens recebidas para o tipo específico de cada uma, em JAVA, enviando-as depois para a fila do RabbitMQ correspondente. A partir desse momento o RabbitMQ encarrega-se de avisar a aplicação da existência de novas mensagens na fila em questão.
- *Outside Translator* – Este módulo ficará à "escuta" de novas mensagens que cheguem às filas do RabbitMQ e irá construir uma mensagem passível de ser enviada através de WAVE para a VANET, para ser recebida posteriormente por outro veículo ou RSU.

Estes dois novos componentes são essenciais para criar uma abstracção entre camada de aplicação e camada de transporte, e vão fornecer todo o suporte necessário para a tradução de mensagens de/para a rede veicular, ver figura 11.

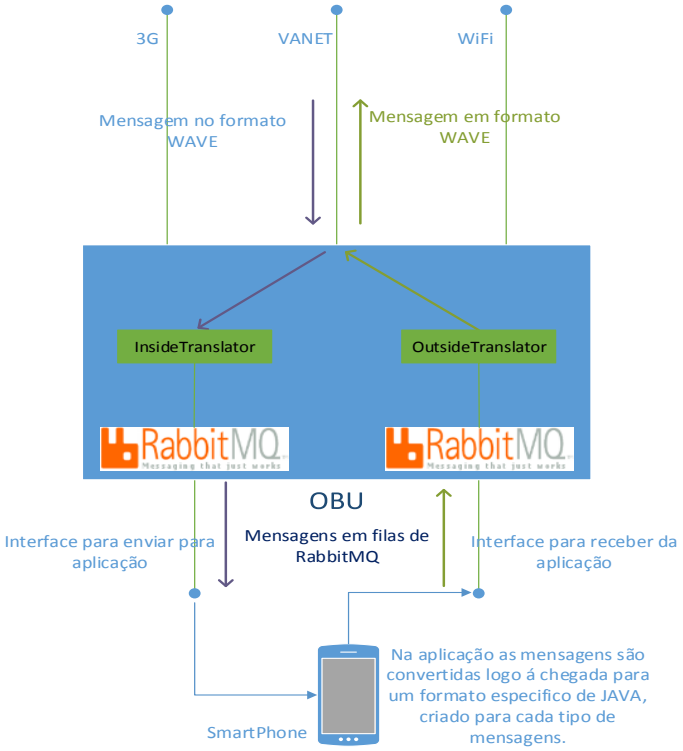


Figura 11 Ilustração do funcionamento dos conversores: Inside Translator – Módulo responsável pela conversão das mensagens recebidas por VANET para classes internas de Java passíveis de envio através do RabbitMQ para o smartphone. Outside Translator – Módulo res Módulo responsável pela conversão da informação recebida no RabbitMQ para mensagens de formato WAVE, para serem enviadas pela VANET.

3.4.2 Acesso ao veículo

Uma das vantagens que o sistema possui é a capacidade de receber alguns dados fornecidos pelo próprio veículo que permitem um melhor controlo e gestão do mesmo. No caso testado, representado na figura 12, os dados eram limitados a consumos de combustível; contudo, o sistema está apto a passar qualquer tipo de dados do veículo, que cheguem pela OBD, desde a OBU até á camada de aplicação. A OBU está sempre a "escuta" de novos valores na interface que possui com o veículo. Quando é recebido um novo valor, de imediato a OBU deve colocar esse valor na fila de envio de mensagens específica, e informar a aplicação que novas mensagens de OBD estão disponíveis para leitura na fila. A aplicação retira a mensagem da fila e analisa os dados recebidos, guardando numa base de dados local e mostrando ao utilizador quando necessário. Contudo deve ser referenciado que a ligação entre OBU e OBD não faz parte do âmbito deste projecto pelo que se fornece suporte para o caso de essa ligação ser existente. Para garantir que estes dados são passiveis de envio e recepção foram criados na OBU simuladores que fornecem os dados simulados a OBU e esta faz o envio dos mesmos para a aplicação MyCar. Para esta comunicação foi necessário colocar serviços de "escuta" a esses simuladores, cujos dados a OBU deve reconhecer como vindos da OBD e enviar o REINVENT

que os encaminha de forma interna para as filas correctas do RabbitMQ. A aplicação fica responsável depois por retirar essas mensagens da fila e tratar de forma conveniente.

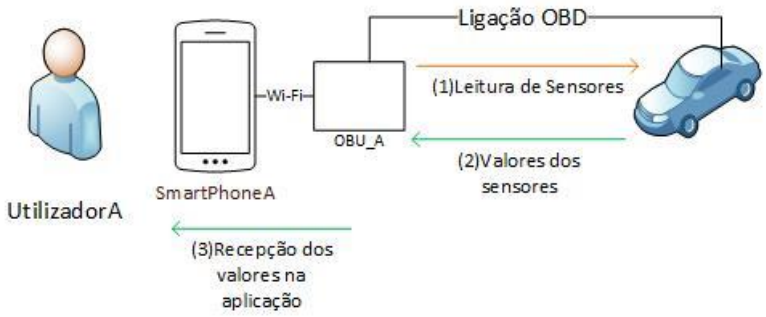


Figura 12 Representação do sistema de mensagens necessário para implementação do caso de uso para acesso a dados do veículo. Esquematizaçao do processo de envio de mensagem de dados dos sensores veiculares até ao *smartphone*.

A seguinte figura 13, representa o mesmo fluxo de informação mas utilizando um diagrama de sequência para o envio de dados dos sensores do veículo para a aplicação. O veículo envia de forma contínua e ininterrupta informação através da OBD dos seus sensores. A OBU está sempre à "escuta" desses dados que a OBD disponibiliza e trata cada mensagem de imediato sem interromper o ciclo de escuta para que nenhuma mensagem se perca. No tratamento dessa mensagem está incluído todo o processo de conversão da mensagem para uma classe de Java interna, envio para a fila do RabbitMQ "CarToPhoneOBD". No telemóvel é lançado pelo RabbitMQ o alerta de nova mensagem existente na fila, e é retirada essa mensagem da fila e tratados os dados conforme o caso (mostrados ao utilizador ou guardados).

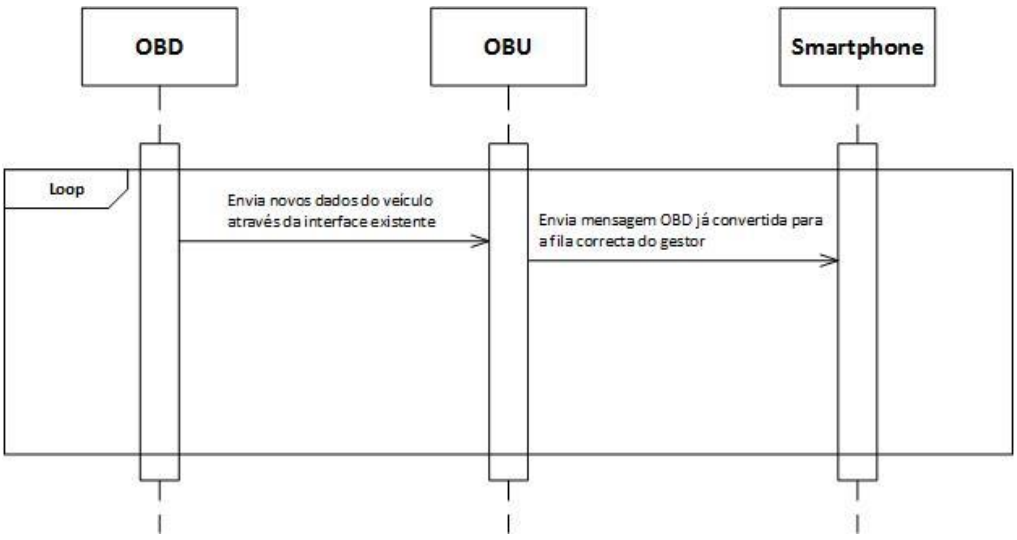


Figura 13 Diagrama de sequência para envio de dados do veículo para o *smartphone* com ilustração dos diferentes componentes utilizados na comunicação entre sensores do veículo e *smartphone*.

3.4.3 Acesso a serviços remotos

O acesso a serviços remotos pode estar muitas vezes limitado devido as características intrínsecas das VANETs. Para possuir um acesso à Internet, caso a ligação de dados (3G ou 4G) esteja desactivada, é necessário que se encontre numa zona de *Hot-Spot* ou no raio de alcance de uma RSU. Por estes motivos é espectável que a ligação possa falhar. No sistema cujo foco foi criar uma aplicação que faça uso de redes veiculares, tentou-se evitar a necessidade de acesso à Internet para a maioria das aplicações. Contudo, funcionalidades como acesso a dados guardados remotamente necessitam indiscutivelmente de ligação à Web.

A OBU quando possui conectividade global, consegue fornecer essa conectividade através da rede Wi-Fi gerada por ela e onde os *devices* dos ocupantes do veículo se ligam para obter ligação a Internet. Assim, para fazer um acesso remoto a serviços pode-se usar as ferramentas próprias do sistema Android que permitem fazer os pedidos e obter as respectivas respostas como se de uma rede Wi-Fi qualquer se trata-se, ficando completamente abstraído do programador o encaminhamento do pedido da OBU para o exterior.

Foram criados alguns *web-services*, que serão abordados em detalhe mais adiante no trabalho, que visam fornecer dados que se encontram armazenados numa base de dados remota. Esses *web-services* satisfazem todos os pedidos que a aplicação efectua, recebendo os pedidos como pedidos *HTTP*, processando sobre os dados as operações necessárias e retornando em formatos *standard* como JSON os resultados das operações.

Na aplicação o processo de envio de um pedido *HTTP* processa-se do seguinte modo (figura 14):

A aplicação antes de fazer o pedido à Internet deve enviar uma mensagem de StatusRequest para saber se o acesso Web está disponível; Se existir conectividade, a placa deve informar a aplicação móvel que pode fazer o pedido *HTTP*. O pedido deve ser feito usando bibliotecas fornecidas pela plataforma Android, uma vez que o *smartphone* do utilizador se encontra ligado a rede *wireless* fornecida pela placa, fica assim transparente o pedido;

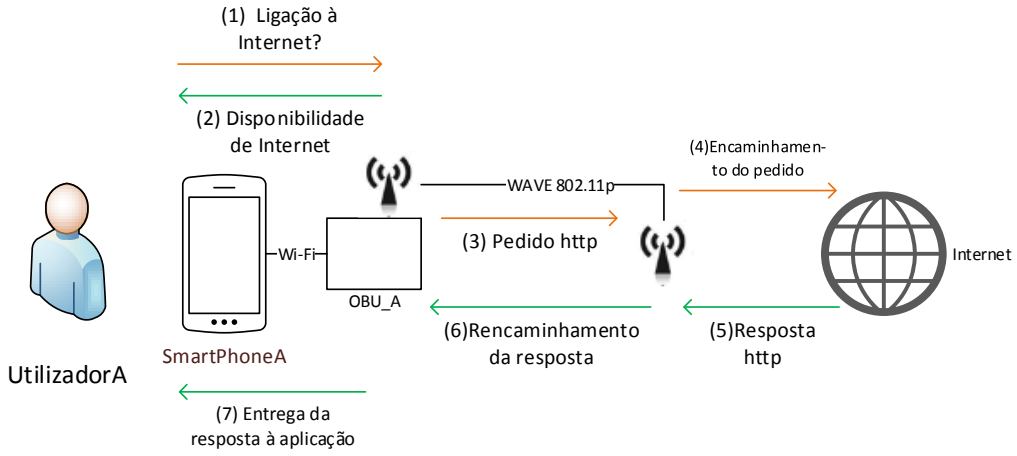


Figura 14 Representação de troca de mensagens necessárias para implementação do caso de uso para acesso a serviços remotos.

Na figura 15 podemos ver como representar o mesmo cenário mas utilizando um diagrama de sequência. Apesar de o diagrama conter no detalhe a OBU, na prática o envio do pedido HTTP da aplicação para o servidor remoto fica abstraído da passagem pela placa veicular pois do lado do *smartphone* apenas importa que o pedido siga pela rede *wi-fi* à qual está ligado.

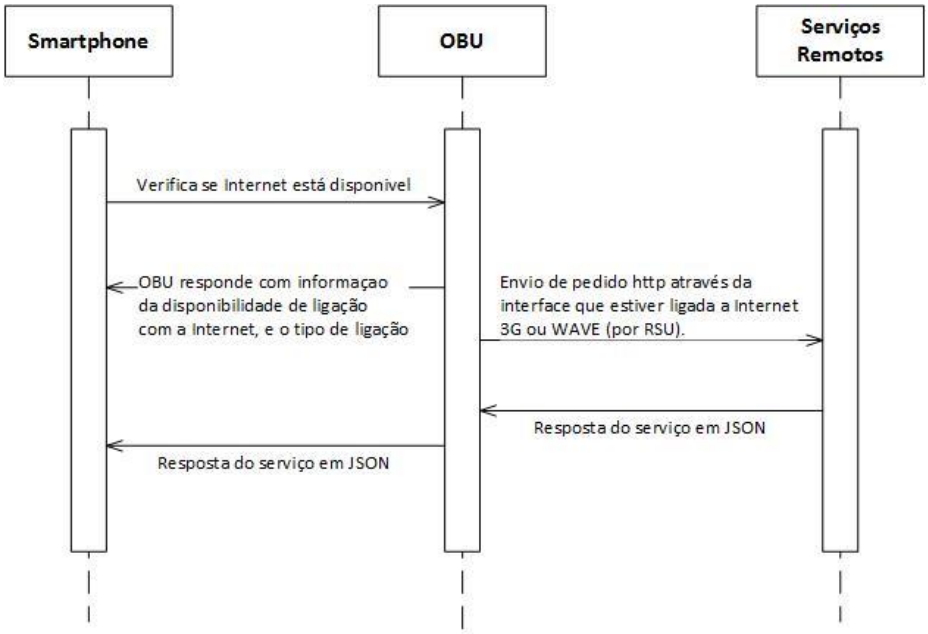


Figura 15 Diagrama de sequência para acesso a serviços remotos pelo *smartphone*

3.4.4 Gateway para pedidos externos HTTP

Na rede veicular o acesso à Internet de modo constante é complicado de manter se a cobertura das RSUs não for suficiente. A solução por Wi-Fi funciona bem em casos pontuais em que o carro circula a baixa velocidade, ou se encontra parado junto de zonas de acesso *wireless* como *hotspots.* A solução de 3G, apesar de ser a mais simples, é a mais dispendiosa pois o tráfego usado é pago.

Uma das grandes vantagens e inovações que este projecto oferece é a possibilidade de enviar através de outro automóvel, informação para a Internet, como ilustra a figura 16. No caso de a OBU responder negativamente à aplicação quando inquirida sobre a disponibilidade de acesso web, a aplicação pode enviar o pedido para a VANET de modo que a mensagem seja recebida por outra OBU. Para isso é necessário que seja implementado nas OBUs *software* que funcione como *gateway* das mensagens de rede veicular para pedidos HTTP. A OBU deve verificar se a mensagem que recebe na interface WAVE é um pedido HTTP de outro veículo; deve depois verificar se possui acesso á Internet através de qualquer uma das interfaces; se existir deve reencaminhar esse pedido para a Web, se não existir deve descartar a mensagem.

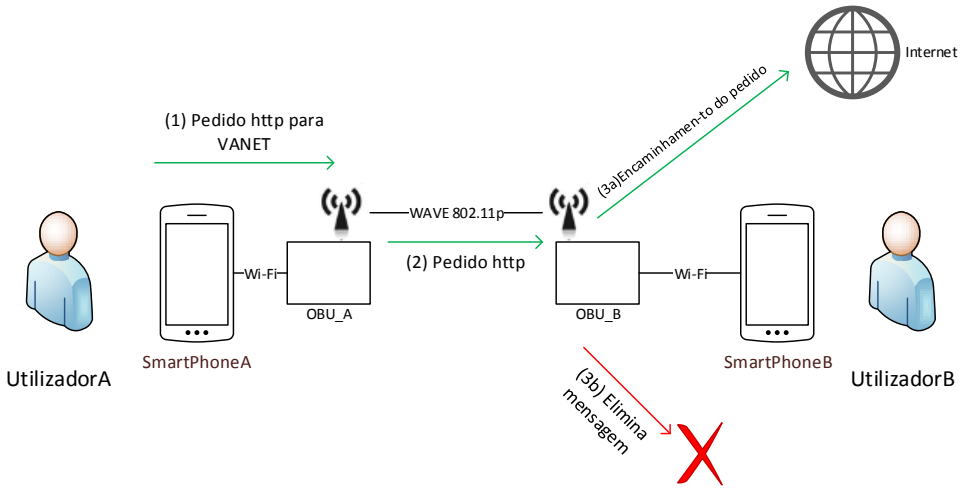


Figura 16 Representação de implementação da Gateway para pedidos HTTP de terceiros.

Para que esta funcionalidade exista, foi necessário adicionar algum código aos módulos da OBU. Para a aplicação nada altera, uma vez que o envio do pedido é feito, como já foi referido, como se de um qualquer pedido a rede Wi-Fi onde o *smartphone* está ligado se tratasse. A única alteração que na aplicação pode existir é a questão de decidir quais os pedidos que podem ser ou não forçados a seguir pela VANET quando a OBU à qual o *smartphone* está ligado não possui conectividade á Internet.

3.5 Sumário

No capítulo 3, foi introduzido o STRIVE. Foram abordados os conceitos teóricos necessários para uma posterior implementação, como arquitectura e modelação.

Foi introduzido o REINVENT, um sistema de troca de mensagens entre *smartphones* e OBU, já existente que vai ser alterado para que possa ser usado no STRIVE no acesso das mensagens à camada de transporte na rede veicular. Identificaram-se os principais módulos que iriam compor o REINVENT nos componentes necessários do STRIVE, smartphone e OBU.

Para funcionamento do REINVENT utiliza-se uma biblioteca externa de um sistema chamado RabbitMQ.É um gestor de filas de mensagens utilizado nas comunicações, por troca de mensagens, entre módulos do REINVENT o que ajuda a criar a camada de abstracção pretendida.

Foram identificados os principais cenários de utilização do STRIVE e com os cenários foram detectados os principais obstáculos, que se agruparam em 3 grupos: comunicação com sensores do veículo; comunicação dentro da rede; comunicação em escala global através do acesso a Internet. Foi ainda identificado um outro grupo de comunicações intrínseco da comunicação na VANET que pretende criar uma *gateway* das mensagens da rede veicular para a Internet aumentando o acesso a recursos externos.

4.Implementação do STRIVE

Neste capítulo descreve-se em maior detalhe as implementações necessárias para a criação do STRIVE. As implementações nos vários componentes visam dar suporte aos cenários descritos previamente, assim como a lista de funcionalidades para o utilizador.

4.1 Extensões no REINVENT

Como *front-end* de todo o sistema era necessário usar um dispositivo que processasse dados e não apenas os mostrasse. Como já foi referido, no mundo actual os telemóveis estão a cair em desuso e vieram dar lugar a telefones "inteligentes", os chamados *smartphones*, que correndo diferentes tipos de sistemas operativos conseguem ter desempenhos semelhantes a computadores pessoais de há uns anos atrás. É fácil hoje em dia encontrar à venda estes *smartphones*, de gama média, com processadores de dois ou mais núcleos a trabalhar a frequências elevadíssimas, tudo graças à evolução da nanotecnologia que permite colocar em aparelhos de dimensões reduzidas o que existe em computadores de escritório ou portáteis.

A escolha da plataforma Android, de entre outras disponíveis como iOS ou Windows Phone, prendeu-se principalmente com a experiência que já existia com este sistema operativo, o que permitiu ultrapassar certas barreiras com maior facilidade do que seria com um SO desconhecido. Além deste factor, o facto de Android ser um sistema operativo aberto que pode ser usado em diversos aparelhos, torna mais abrangente o público-alvo e facilita também no processo de testes à aplicação.

De seguida será explicado em traços gerais como foram implementados os principais módulos que compõem a aplicação.

4.1.1 Arquitectura Android

A arquitectura geral de Android, como pode ser observado na figura 17, assenta num núcleo de Linux, camada de mais baixo nível onde estão os controladores que dão acesso aos recursos físicos do telemóvel. Estes recursos não são, normalmente, acedidos directamente pelas aplicações, devido a questões de segurança, mas através de bibliotecas que fazem a gestão de acesso das várias aplicações a esses mesmos recursos. Essas bibliotecas dão ainda lugar a camadas de mais alto nível, *frameworks*, que juntam diversas bibliotecas para efectuar diversas operações usadas pela maior parte das aplicações.

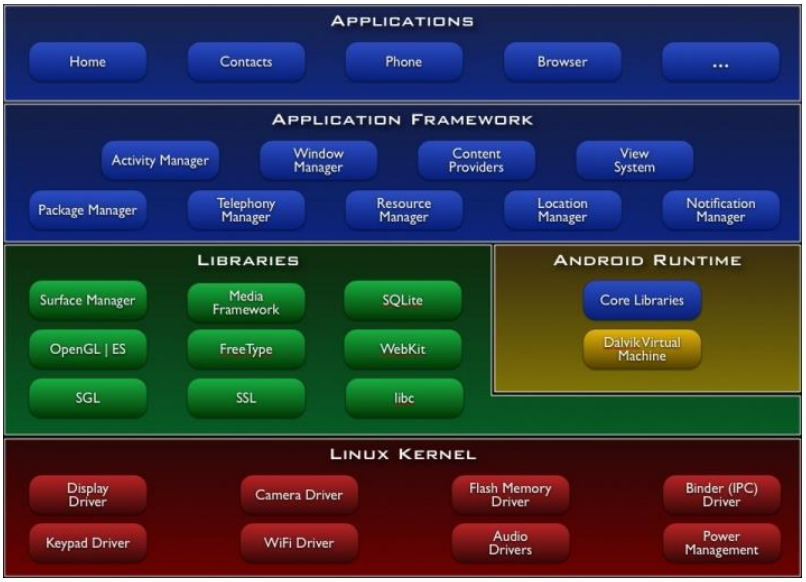


Figura 17 Arquitectura do Sistema Operativo Android, com visualização dos vários componentes internos, nomeadamente Content Providers no qual se baseia o funcionamento do REINVENT.[9]

No STRIVE usamos um *Content Provider* (*application framework)* criado pelo REINVENT. Servirão de suporte ao sistema de alerta de recepção e envio de mensagens via RabbitMQ. Utilizamos Serviços ("service") do Android para implementar o sistema de escuta e processamento de novos alertas emitidos pelo REINVENT (*content provider)*.

Usamos SQLite (base de dados relacional) para assegurar a persistência dos dados localmente no telemóvel nomeadamente dados vindos da placa como GPS e consumos. Esta informação, após processamento é apresentada aos utilizadores. Além destas *frameworks*, outras de uso mais comum como *Activity, Activity Manager, Fragment e Views* foram usadas para criar os ecrãs que permitem a visualização dos dados.

4.1.2 Content Providers

Os *Content providers (provider)* são uma abstracção baseada numa arquitectura REST que permite gerir acessos e partilhar infromação de uma fonte de dados "lógica" entre diferentes aplicações em Android usando uma designação única para lhes aceder - um Uniform Resource Identifier (URI). O *content* p*rovider* (instância da classe que representa o *Content Provider)* recebe pedidos de dados dos clientes, resolve esse pedido, e devolve o resultado (dados pretendidos).

A utilização de um *Content Provider* implica o recurso a um *Content Resolver (resolver)* que e irá mapear no cliente a informação que o *provider* acede.

O *resolver* permite que se crie uma escuta para alterações que o *provider* faça a um determinado recurso. Deste modo quando existe uma alteração e.g. nova mensagem, o *provider* vai lançar uma notificação para o URI identificador do tipo de mensagem recebido. Todos os tipos de mensagens

ficam registados nos respectivos URIs (um por cada tipo) no *provider*. Assim o *resolver* irá ser informado de qualquer nova alteração que ocorra nos URIs que estiver à escuta.

O REINVENT suporta todo o processo de gestão de mensagens por meio de um *Content Provider* acessível para várias aplicações no mesmo dispositivo.

4.1.3 Services

Os serviços ("services") na plataforma Android são uma solução que permite a uma aplicação realizar determinadas operações de longa duração que não interagem com o utilizador, ou seja que executam em plano de fundo.

No projecto desenvolvido quer na aplicação *MyCar*, quer no REINVENT instalado no Android (*content provider*), foram utilizados vários serviços que para verificar a ocorrência de novas mensagens da OBU, e efectuar processamento sobre informação recebida.

Por exemplo, no caso do GPS, figura 18, os serviços suportam um fluxo de processamento que permite difundir e processar os alertas de novos dados desde a OBU até ao *smartphone e* aplicações cliente. Associando os *Servi*ços ao *Content Provider* responsáveis pelo acesso a cada tipo de mensagens, é possível verificar em background as alterações lançadas pelo *provider* e despoletar as operações necessárias que podem ser desde guardar numa base de dados local até ao simples lançar de uma nova notificação de " nova mensagem já tratada" que vai ser escutada pela aplicação e mostrada ao utilizador.

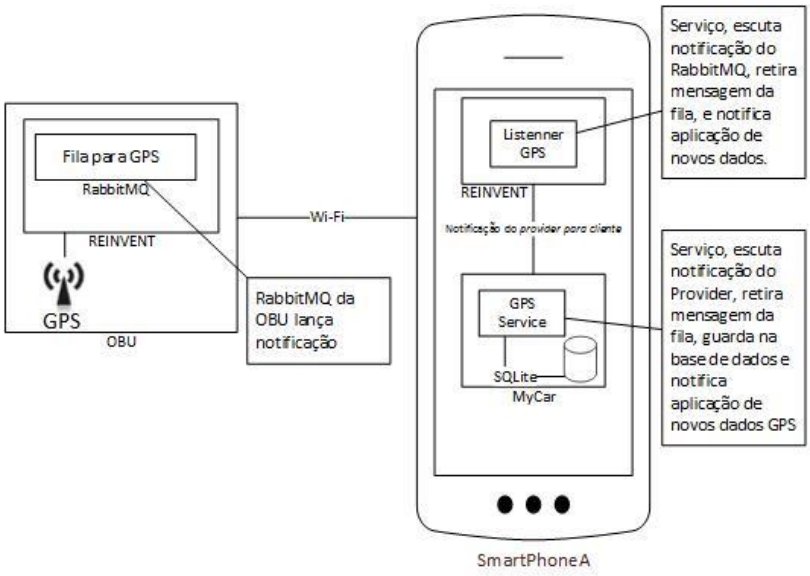


Figura 18 Ilustração do sistema de alertas

4.1.4 Implementação do REINVENT na OBU

Na OBU o módulo do REINVENT tem diversas funções que permitem que a OBU detecte a existência de novas mensagens nas várias interfaces e faça o encaminhamento para a interface correcta

de saída, uma vez que não existe elevado processamento de dados na OBU além da detecção do tipo de mensagens recebidas.

Recorrendo a um programa criado em *Python,* são inicializados os processos de "escuta" das interfaces (VANET, Wi-Fi, OBD). Quando se dá a chegada de uma mensagem, para cada interface, existe um módulo "*Client.py"* complementar (também em *Python*) que faz o processamento dessa mensagem na medida em que identifica que tipo de mensagem foi recebida e para onde deve ser encaminhada.

Analisando as mensagens recebidas pela VANET, temos que todas as mensagens são para ser reencaminhadas para o RabbitMQ. O REINVENT, após detectar o tipo de mensagem no *Insidetranslator,* faz a passagem da mensagem a uma função que se encarrega de colocar a mensagem na fila correcta do RabbitMQ que lança a aplicação avisando o RabbitMQ do *provider* da existência de uma nova mensagem.

De forma análoga acontece para as mensagens que chegam à interface Wi-Fi vindas da aplicação através do RabbitMQ. Existem mensagens como o caso do StatusRequest, que serve para obter dados de conectividade de OBU que devem ser tratadas pela OBU. A OBU detecta o tipo de mensagem recebida e adquire as informações necessárias para a criação de uma mensagem de StatusResponse, e encaminha-a para o RabbitMQ na fila própria de resposta à aplicação.

Excepto alguns casos pontuais, todas as outras mensagens devem ser apenas reencaminhadas para a interface de VANET para que cheguem aos outros veículos ou RSUs.

4.1.5 Implementação do REINVENT no Android

O REINVENT como já foi abordado é composto por vários módulos, um deles é uma aplicação Android que serve de suporte a outras aplicações.

Para servir de suporte, esse módulo trabalha como *provider* dos dados que recebe do RabbitMQ ou seja das mensagens. É o único módulo com acesso ao RabbitMQ, ou seja com acesso imediato às mensagens, garantindo assim que as mesmas são recebidas e tratadas da forma correcta e segura.

O REINVENT é composto de duas classes principais, a classe *NetworkProvider.java,* e a classe *MessageListenerService.java.*

A classe *MessageListenerService.java* é uma extensão da *framework Intent Service* e é inicializada pela classe *NetworkProvider.java.* Nesta classe são iniciadas 10 *threads* responsáveis pela recepção e processamento de novas mensagens. Em cada uma das *threads* é estabelecida a ligação ao servidor de RABBITMQ que está instalado na OBU, através dos métodos "ConnectToRabbitMQ_" sendo que cada diferente terminação do nome do método corresponde a ligação uma fila específica do RabbitMQ.

Após essa ligação ser estabelecida, é iniciada uma escuta a novas mensagens. O alerta de uma nova mensagem na fila é assegurado pelo RabbitMQ, e apenas é necessário efectuar o "consumo" dessa mensagem.

Cada tipo de mensagem é convertido para a classe JAVA correspondente, e é depois inserido numa estrutura, e é lancada uma notificação ("*notifyChange()"*) através do *Content Resolver*, de uma alteração no URI correspondente ao tipo de mensagem. Essa notificação vai ser depois detectada pela aplicação, como será explicado mais adiante neste documento.

É deste modo que é feita a escuta e aviso de recepção de novas mensagens enviadas pela OBU para o *smartphone*.

Ainda no contexto do REINVENT, vamos abordar a classe *NetworkProvider.java*, figura 19. Esta classe é responsável por criar uma interface para as aplicações obterem novas mensagens enviadas pela OBU, e é também responsável por criar os métodos necessários para o envio de mensagens do *smartphone* para a VANET.

A classe possui vários métodos de interesse a quem pretende fazer uso da API, contudo por questões de legibilidade apenas serão apresentados e comentados os principais, descritos em mais detalhe no ANEXO A.

Métodos mais importantes:

- *call(String URI, String method, String args, Bundle extras) :*

Este método é utilizado pela aplicação quando quer obter uma mensagem nova, imediatamente a seguir a obter uma notificação lançada pela classe *MessageListenerService.java* (descrito previamente). Para facilitar o uso do método por parte de terceiros, foi criado uma classe auxiliar, *StaticUtils.java*, que contém as *strings* que devem ser usadas nos argumentos de entrada do método.

Exemplo de utilização para envio de mensagem de texto:

```
      "Bundle bdl = getContentResolver().call(
      StaticUtils.URIProvider,
StaticUtils.CallGetTXTMessageMethod,
StaticUtils.CallGetTXTMessageArgs,null);"
```

- *insert(URI, Content Values values):*

Este método é utilizado pela aplicação quando quer enviar dados para o *provider* para serem enviados para a VANET. Este método vai ficar encarregue de receber os dados da aplicação, construir as mensagens dos vários tipos de forma conveniente, e proceder no fim ao seu envio para a rede veicular.

Exemplo de utilização para envio de alerta de veículo:

Criar o dicionário com as chaves e os valores:

```
“final ContentValues cv = new ContentValues();
          cv.put("AlertType", alertType);
          cv.put("Description", textOptional);
          cv.put("Problem", searchItem);
          cv.put("Latitude", gpsPoint.getLatitude());
          cv.put("Longitude", gpsPoint.getLongitude());
          cv.put("CarID", StatusStatic.getObu_ID()); “
```

Criar URI para o recurso:

```
“final Uri uri = Uri.parse(StaticUtils.URIProvider+
"/newAlertMapmessage");”
```

Envio dos dados para *provider:*

```
“ ContentResolver cres = getActivity().getContentResolver();
 cres.insert(uri, cv); ”
```

Deste modo os dados são enviados para o REINVENT (*provider)* e serão, depois de convertidos para mensagens aceites pelo RabbitMQ, inseridos nas respectivas filas e enviados para a OBU que os encaminha para a VANET ou os trata, conforme o tipo de mensagens.

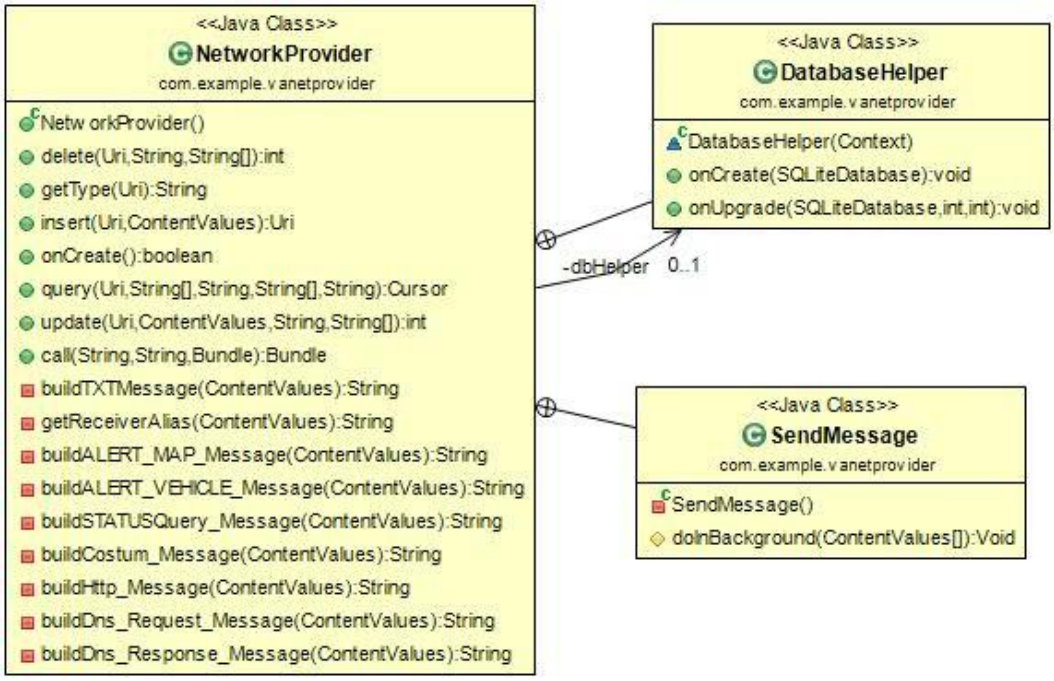


Figura 19 Diagrama de classes do REINVENT, visível a ligação entre classes necessárias ao funciomento do REINVENT para envio de mensagens, e acesso a dados da base de dados interna.

4.1.6 Gateway para VANET

Este caso especifico de mensagens para a rede veicular foi criada uma ligação por gateway para a VANET, tendo em conta o cenário previamente descrito na secção 3.4.4 e visível na figura 16, e

devido às diferenças que apresenta em relação às outras mensagens enviadas para a VANET vai ser descrito em maior pormenor.

Quando a aplicação *MyCar* obtém da OBU uma resposta negativa sobre a conectividade à Internet, é enviado para o módulo REINVENT do *smartphone* uma mensagem do tipo *HTTP* com todos os dados necessários. No REINVENT essa mensagem é colocada na fila própria do RabbitMQ, "PhoneToCarHTTPMSG", que envia essa mensagem para a OBU.

Na OBU essa mensagem é recebida no REINVENT local e é convertida para uma mensagem em formato WAVE, e enviada de seguida para a VANET.

Quando se dá a recepção de uma mensagem do tipo *HTTP* na interface WAVE da OBU, o equipamento deve verificar a sua ligação á Internet. Se existir então encaminha esse pedido para a Internet, e caso a OBU ainda consiga obter uma resposta em tempo útil converte essa resposta para uma mensagem *HTTPResponse* que é encaminhada para a rede veicular para que possa ser recebida pela OBU que a enviou. É necessário que a OBU mantenha um registo dos identificadores dos equipamentos que enviaram para si mensagens de *http* pela VANET para que se possa depois identificar na mensagem *HttpResponse* o destinatário da mesma.

4.2 Serviços Web

Para dar suporte a aplicações, e para questões de gestão e processamento de grandes quantidades de dados, foram criados serviços para lidar de forma simples com os dados da base de dados remota. Esta base de dados contém todos os dados do sistema que são inseridos inicialmente quando o STRIVE é instalado, assim como os dados que as OBUs vão depois enviando durante a utilização.

O diagrama da base de dados na altura da realização deste trabalho era o que podemos visualizar na figura 20. Neste diagrama são visíveis algumas classes (tabelas) de maior importância para o sistema STRIVE. É o caso da tabela "obd", onde serão inseridos dados que o veículo reporta provenientes dos sensores, juntamente com o identificador da OBU que transmite os dados à aplicação, e o instante de envio desses dados da OBU para a aplicação. Na classe "gpspoints", são inseridos os pontos gps (latitude, longitude e altitude) de cada veículo, assim como o identificador do veículo, o instante que a mensagem foi enviada para a aplicação, velocidade de circulação e a precisão da localização. As classes "obd" e "gpspoints" foram criadas posteriormente ao resto da base de dados, daí a incompleta correlação com as restantes classes. Estas tabelas, pelo funcionamento do STRIVE, são actualizadas pela aplicação *MyCar*, que mantém esses dados guardados até ser feita a actualização da base de dados, quando existe uma ligação à Internet disponível. Outras classes usadas pelo STRIVE são o "alertmap" e "alertvehicle". Estas classes contêm informação relativa aos alertas que os utilizadores da aplicação *MyCar* vão relatando para a base de dados remota quando existe acesso à Internet. Essa informação é depois fornecida aos utilizadores do *MyCar* quando a aplicação faz pedidos de alertas relatados por outros utilizadores e que foi guardada remotamente.

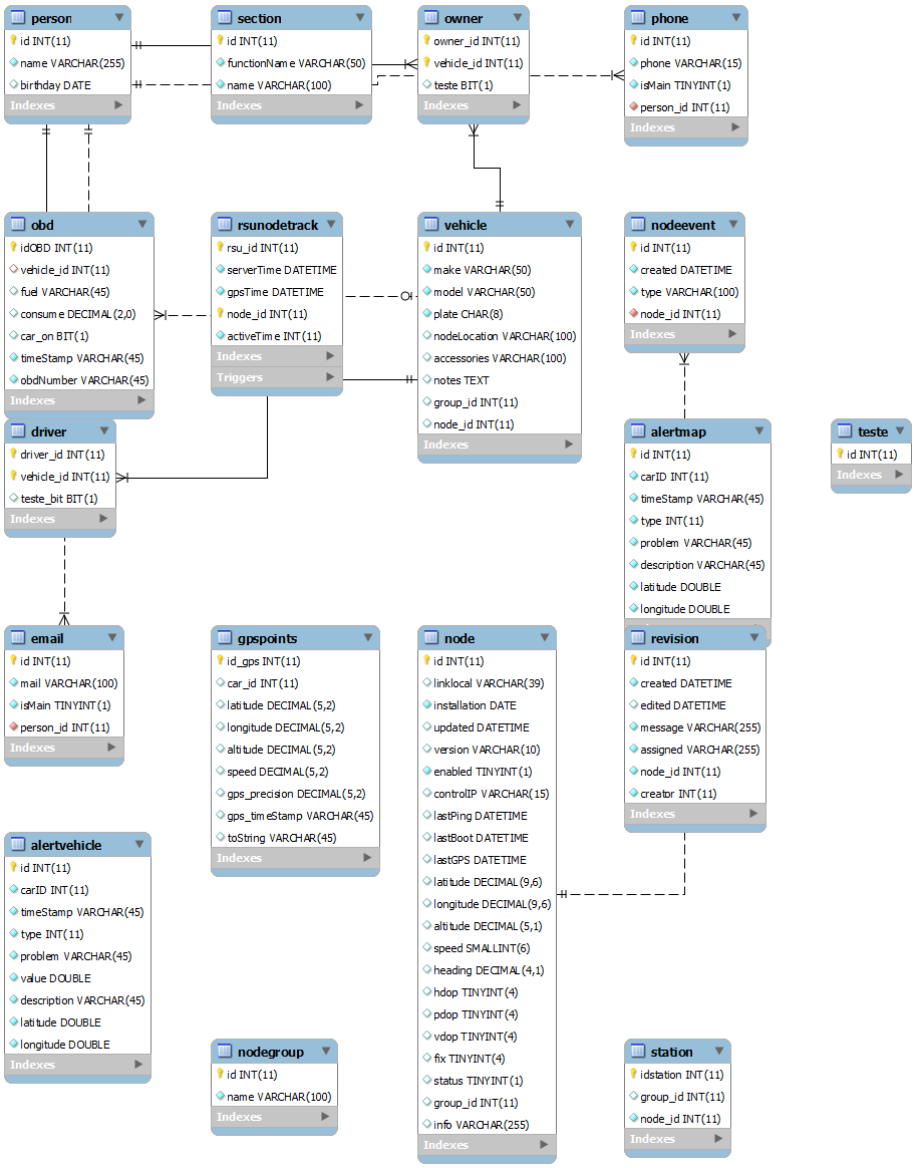


Figura 20 Diagrama de Classes da Base de Dados.

4.2.1 Serviços existentes

Existem muitos dados que podem ser úteis para o utilizador, e muitos outros que devem permanecer "escondidos" da maioria dos utilizadores e apenas gestores do sistema ter acesso. Contudo existem outros que aplicações como o *MyCar* precisam de ter acesso para fornecer mais informações e mais completas ao utilizador. É o caso dos dados detalhados sobre veículos em redor.

Analisando a figura 20, vemos que na tabela *node* existem alguns dados relacionados com os veículos que podem ser fornecidos e outros que não são visíveis. A um condutor pode interessar saber a posição GPS dos outros veículos para que possa saber quem se encontra em seu redor, assim como a velocidade a que circula, ou até a matrícula que pertence a tabela *vehicle*. Contudo dados como versão da OBU, data de instalação, informação técnica, não devem estar visíveis.

Foram criados *web services* para todas as tabelas da base de dados, que permitem obter todos os dados de cada tabela, remover uma determinada entrada da tabela e editar uma entrada.

Para efectuar essas operações o método é comum a todos os serviços. A interacção com o serviço faz-se através de pedidos HTTP colocando no corpo da mensagem os dados que se quer inserir em JSON. Se o utilizador quiser adicionar uma nova entrada à base de dados deve efectuar um pedido do tipo *Post* com o url para a tabela em questão. A aplicação MyCar faz 4 tipos de inserções na base de dados remota: alertas de mapa, alertas de veículo, dados de GPS, dados de OBD.

Se for o caso de querer editar uma determinada entrada de uma tabela deve-se fazer uso do pedido *Put* em que os dados enviados em JSON vão ser usados para substituir dados existentes.

Para se eliminar uma determinada entrada deve-se utilizar o método *Delete*. Para identificar a entrada que se pretende eliminar, no url do serviço deve ser acrescentado um valor inteiro no fim do url, que indica qual o id da entrada que se pretende apagar na tabela.

Para se obter uma lista de todas as entradas da tabela, deve ser utilizado o método *Get* no pedido HTTP. A informação de resposta vem como um *array* de dados em JSON.

Para análise tomaremos como exemplo um *web-service* para inserção na Base de Dados e outro para leitura.

4.2.1.1 Web-Service *para inserção na Base de Dados*

Como exemplo, tome-se a inserção por parte da aplicação de novos alertas de problemas de mapa enviados remotamente pelo condutor/ocupante através da aplicação *MyCar*.

Na aplicação para efectuar o envio de pedidos para a base de dados faz-se uso da classe *Parser.java*. Esta classe tem alguns métodos públicos para enviar os dados através da Internet. No caso de se querer enviar alertas do veículo, existe o método *setAlertMap()* que permite o envio de um alerta especifico para a tabela de alertas de mapas da base de dados (tabela *alertmap*) . O método recebe nos argumentos toda a informação (vinda do utilizador e da própria aplicação) que é necessário para preencher uma entrada da tabela. Os dados recebidos, são convertidos numa estrutura de nome *AlertMap*, para que possa com ajuda de uma biblioteca externa, *Gson* [31], ser convertida para um objecto JSON que é depois enviado para o url do *web-service.*

Dentro do método o código seguinte é responsável por efectuar um pedido de *POST* de *HTTP*:

```
AlertMap alert = new AlertMap(id, carID, timestamp, type, problem,description, lat, lon);

HttpClient httpClient = new DefaultHttpClient();
HttpPost httpPost = new HttpPost("URL_WEB_SERVICE_ALERT_MAP");

Gson gson = new Gson();
StringEntity e = new StringEntity(gson.toJson(alert),HTTP.UTF_8);
e.setContentType("application/json");
httpPost.setEntity(e);

HttpResponse response = httpClient.execute(httpPost);
```

O *Post* é recebido pelo servidor e tratado em conformidade com o url inserido. Neste caso exemplificado o url deverá ser:

"**ipServidor:8080/Vannet_WebServices/webresources/com.mycompany.webservices.alertmap**".

Do lado do servidor existe uma definição para o método *post*:

```
@POST
  @Override
  @Consumes({"application/json"})

  public void create(Alertmap entity) {
     super.create(entity);
  }
```

O servidor "consome" a informação enviada em JSON no pedido. Essa informação, é convertida para uma estrutura interna semelhante á estrutura presente na aplicação, previamente mencionada AlertMap.java e que no servidor ganha o mesmo nome.

O servidor onde foram desenvolvidos os serviços, está ligado ao servidor onde está instalada a base de dados desenvolvida, deste modo, usando as ferramentas certas, torna-se possível efectuar de forma rápida, do lado do servidor dos serviços, as pesquisas, inserções e remoções.

Foram criadas estruturas no servidor dos serviços para cada representar cada tabela, com recurso a anotações de *Persistence* [32]. Estas anotações permitem que de forma simples se consiga associar a classe à tabela de modo que a inserção se faça no servidor de modo automático para a base de dados.

4.2.1.1 Web-Service para obter dados da Base de Dados

De um modo análogo ao que foi explicado para o caso do *post*, a aplicação quando pretende obter dados da base de dados central deve ser feito um pedido *HTTP* em que o método é *Get.* Ao aceder a um url com um pedido de *Get* o serviço irá retornar todas as entradas desse tipo de recurso, caso no url não exista um parâmetro de pesquisa.

Um exemplo de serviço que o serviço utiliza é o serviço para obter as posições dos veículos através da base de dados.

O serviço retorna uma lista com informação detalhada sobre cada veículo, como matrícula, latitude, longitude e altitude da sua localização.

Para obter essa lista é necessário o seguinte código:

```
HttpClient httpClientGet = new DefaultHttpClient();
HttpGet httpGet = new HttpGet(url);
HttpResponse httpResponse = httpClientGet.execute(httpGet);
```

No campo "url" deve estar o *link* para acesso ao servidor que trata o pedido HTTP. Neste caso específico, este serviço é garantido por um servidor diferente do usado no caso do *Post*. Todos os serviços foram desenvolvidos de raiz para fornecer suporte à aplicação, contudo o caso da lista de

veículos, era um serviço já existente o que implica um url diferente, **"http://vtestbed.nap.av.it.pt/controller/?user&op=login&type=json".**

4.3 Aplicação MyCar

Como *proof of concept* de todo o sistema criado foi desenvolvida uma aplicação que implementasse a lista de funcionalidades, enumerada previamente, que permitisse trazer as principais vantagens das VANETs para o utilizador comum.

Vamos analisar os vários ecrãs da aplicação e fazer a relação entre estes e as funcionalidades que oferecem.

No ecrã inicial da aplicação é pedido ao utilizador o seu nome. Esse nome é o que fica associado à aplicação naquela sessão e vai ser usado: quando é enviada uma mensagem para outro utilizador para que o receptor saiba quem enviou a mensagem; quando a aplicação recebe uma mensagem para verificar se o nome de destinatário é o mesmo que o receptor; para "registar" na aplicação e no DNS o utilizador (o DNS foi um trabalho fora do âmbito do trabalho aqui descrito).

Após o ecrã inicial o utilizador tem à sua disposição uma barra com 5 separadores, cada um com um ecrã diferente que lhe dá acesso a diferentes funcionalidades.

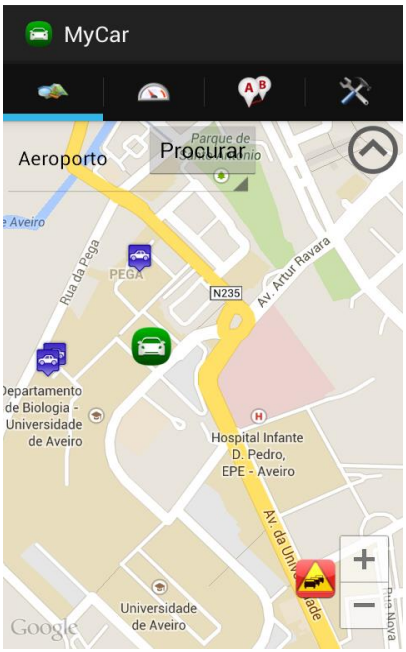


Figura 21 Ecrã com mapa e localizações de utilizador, outros veículos e avisos.

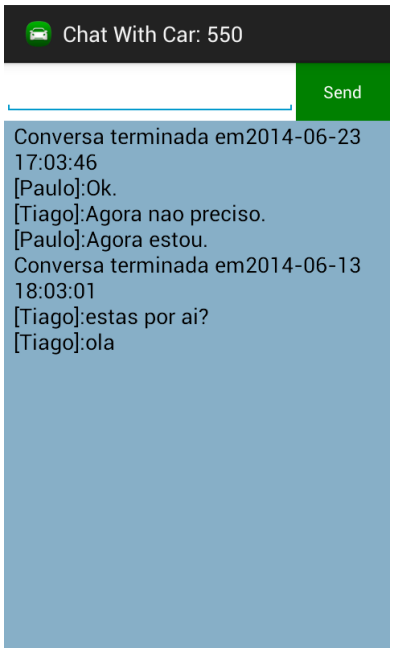


Figura 22 Comunicação por texto entre utilizadores.

O separador escolhido para abrir de imediato, seguindo uma ordem lógica da esquerda para a direita é o ecrã com o mapa, figura 21. Neste ecrã, o utilizador pode observar a sua posição, no mapa fazendo uso do serviço gratuito de mapas, *Google Maps*, o que lhe permite saber onde se encontra em cada instante. Nesse mapa estão também visíveis as posições dos veículos equipados com OBU que se encontram num raio, predefinido de 500 metros. Caso exista ligação ao servidor remoto, a aplicação

tenta obter através de um *web service* os últimos alertas de mapas enviados para o servidor e mostra esses alertas no ecrã do utilizador como é visível na figura 21 pelo símbolo vermelho e laranja, exemplo de um alerta de trânsito. É neste ecrã que irão também aparecer erros relacionados com veículos em redor recebidos pela VANET ou recebidos da Internet através de um serviço *web.*

Quando se clica no símbolo de um veículo a nossa volta, abre um novo ecrã com vários detalhes sobre o veículo seleccionado como Identificador, Matrícula, Velocidade de Circulação, Posição GPS, Ultima actualização de dados. Estes dados detalhados do veículo, só estão disponíveis quando existe uma ligação ao servidor remoto onde está a base de dados central, pelo que podem não estar sempre disponíveis (como no caso de não haver ligação à Internet). Ainda neste ecrã está disponível um botão, "Conversa", que permite ao utilizador ter acesso a um novo ecrã onde poderá ter uma conversação com o utilizador, figura 22. Graças ao trabalho de DNS desenvolvido em paralelo o utilizador poderá ter uma conversa com outro utilizador que deseja independentemente do veículo onde circule. Contudo existem algumas limitações que ainda não foram ultrapassadas, como o caso de vários utilizadores colocarem nomes iguais no DNS.

Contudo, até à data da escrita da Dissertação não foi possível uma junção dos dois trabalhos, mantendo deste modo a conversação com cada veículo.

A aplicação mantém um histórico de conversações mantidas pelo utilizador, contudo devido a uma incapacidade do sistema não é possível agrupar essas conversas através do destinatário, mas apenas com o veículo destinatário. Essa questão prende-se com a dificuldade de identificar de forma única e inequívoca o utilizador destino em sessões diferentes.

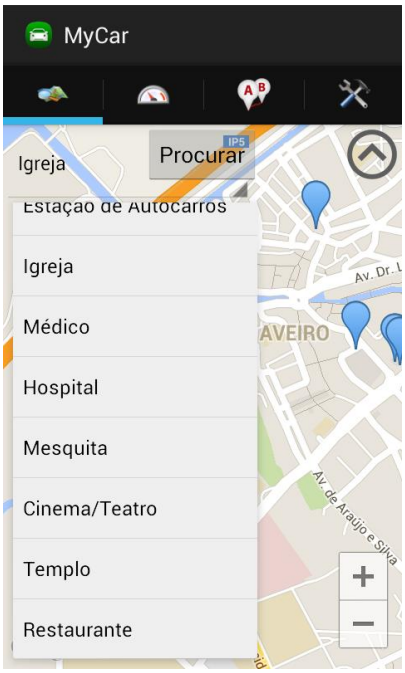


Figura 23 Ecrã com lista de pesquisas de interesse.

Figura 24 Ecrã com exemplo de resultado de local de interesse.

No ecrã com o mapa, visível na figura 23, é também possível fazer uma pesquisa por locais de interesse nas proximidades, através do serviço *Google Places*, que mostra ao utilizador ícones azuis que representam os locais pesquisados. Estes ícones quando pressionados mostram informação útil disponível sobre o local assim como uma foto, quando disponível como está representado na figura 24. Este serviço de pesquisa de locais de interesse prossupõem uma ligação à Internet, pelo que pode não estar sempre disponível.

Focando agora a atenção na barra com os separadores, clicando no segundo separador o utilizador terá acesso ao ecrã onde aparecem os dados em tempo real que a OBU lhe vai dando sobre o veículo, ou seja, velocidade e consumos, visivel na figura 25. Num futuro poderá ser alargado o tipo de dados devolvidos pelo veículo conforme o que o veículo permitir ter acesso, como actuadores dos travões, indicadores de mudança de direcção, pressão dos pneus etc.

Fazendo uso de uma base de dados interna a aplicação mantém um registo dos dados que recebe do veículo (OBD) e dados de GPS da antena da OBU. Através de um sistema de sincronização de dados a aplicação tenta associar a cada ponto GPS recebido os dados OBD que chegaram no mesmo instante. Desta forma, a aplicação consegue dar informações ao condutor como distância percorrida desde o início da viagem, e o consumo médio do veículo.

Figura 25 Ecrã para visualizar dados do veículo.

Se na barra de separadores escolhermos o terceiro separador, iremos para uma lista que contém as rotas efectuadas pelo veículo nos últimos dias, figura 26. Para efeitos de teste de usabilidade foi escolhido como espaço temporal 5 dias, ou seja, as rotas que o carro efectuou cujo ponto de início tem até 5 dias, são mantidas pela aplicação e mostradas nesta lista. A aplicação, todas as vezes que inicia tenta fazer uma ligação à base de dados de modo a tentar "depositar" toda a informação acumulada

sobre o veículo (OBD) e dados da OBU, como GPS. Deste modo fica disponível um acesso aos dados por parte de gestores de frota ou mesmo do utilizador mas visível em *website* em qualquer lado.

Neste ecrã as rotas irão aparecer em lista, mostrando em cada entrada da lista a data e hora de início da viagem. Seleccionando uma das rotas, o utilizador é levado para um ecrã com um mapa e a sua rota traçada sob ele, como mostra a figura 27. Essa rota é construída unindo os vários pontos GPS que foram recebidos. Esses pontos são visíveis no mapa através de marcadores vermelhos, cada um deles seleccionável, e quando se clica é mostrado um ecrã com detalhes do veículo naquele ponto preciso, visível na figura 28.

Além destes dados, foi implementado um sistema de feedback das zonas de maior consumo de forma simples e visual sem o utilizador ser obrigado a verificar os pontos um a um, figura 27. Entre 2 marcadores vermelhos, que representa um ponto GPS recebido, é traçada uma linha. As linhas que desenham as rotas no mapa são coloridas conforme o gasto médio entre esses dois pontos que ficam nas extremidades de cada linha, que representa uma parcela da rota. Segundo um código simples de 3 cores (vermelho, amarelo, verde), associado respectivamente a consumo médio superior a 10L/100km, consumo entre 7.0L/100km e 10.0L/100km, e consumo menor que 7.0L/100km, e com recurso a uma legenda, fica de fácil percepção as zonas de maior e menor consumo.

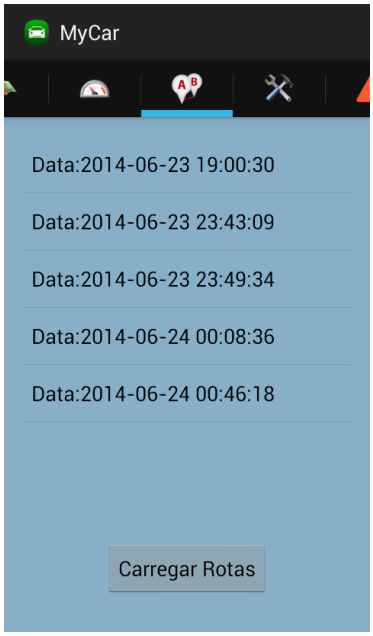


Figura 26 Ecrã com lista de rotas efectuadas pelo veículo.

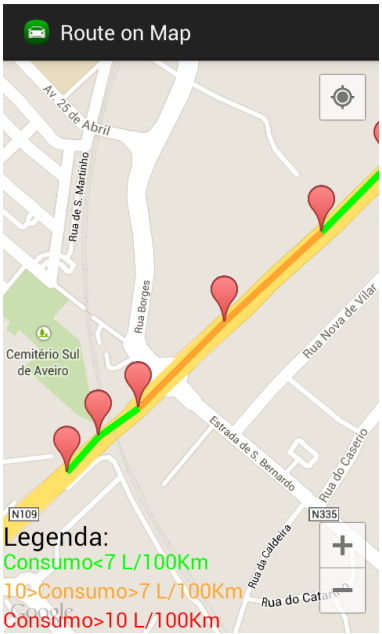


Figura 27 Ecrã com rota em detalhe selecionada.

Figura 28 Ecrã com detalhes de um ponto específico da rota.

Voltando ao ecrã das rotas, e escolhendo o quarto separador, o utilizador é "levado" para um ecrã, figura 29, onde dispõem de uma lista com problemas associados ao veículo. Em cada entrada é visível o tipo de problema que se trata e o número de alertas de problemas desse tipo relatados pela OBD através da OBU. Este número funciona como sistema de avaliação para o utilizador que tem deste

modo um feedback do estado do veículo. Além deste relatório visual de várias partes do veículo é possível ao utilizador lançar ele próprio um alerta sobre um problema por ele detectado.

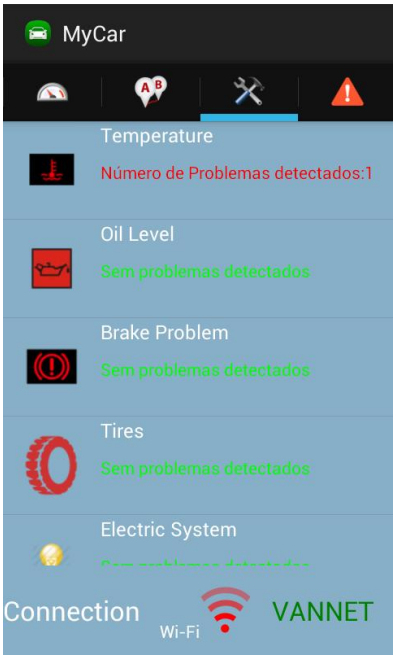


Figura 29 Ecrã com lista de detalhes técnicos do veículo.

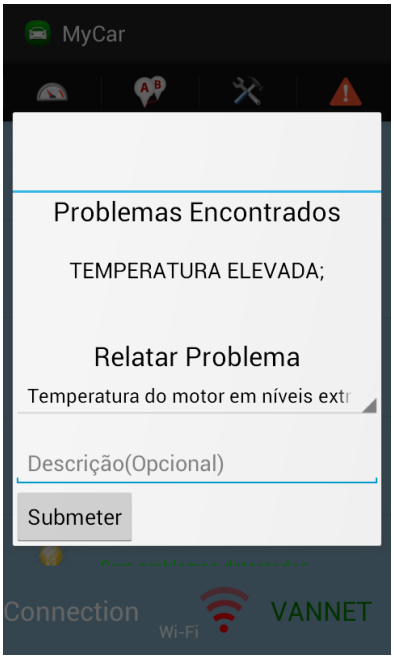


Figura 30 Ecrã para seleccionar dados sobre o problema do veículo para enviar via VANET e via Web.

Através do clique, escolhe o tipo de erro em que se encaixa o problema detectado. Depois, surge um ecrã onde é possível escolher o erro de alguns pré-definidos ou então acrescentar um novo, figura 30. Neste ecrã, caso existam erros relatados pela OBD irão aparecer as descrições dos mesmos, para que o utilizador saiba o tipo de erros existentes antes de fazer o envio de alerta. Por fim existe um botão para enviar esse relatório quer para a rede veicular de modo a avisar os condutores em redor que o veículo circula com problemas ou até mesmo uma avaria e paragem, quer para a Internet (se existir acesso) de modo a informar a base de dados remota do problema do veículo. Esta funcionalidade torna-se particularmente útil se for um cenário de gestão de frotas para que exista um maior e melhor controlo do estado dos veículos.

Por último, acedendo ao quinto separador da aplicação é mostrado um ecrã que tem um funcionamento idêntico ao ecrã com listagem de erros do veículo. Existem 4 botões que identificam 4 tipos de alertas relacionados com o mapa que o utilizador pode enviar, figura 31. Estes alertas são enviados através da rede veicular para outros veículos em redor, e assim como no caso dos alertas relacionados com o veículo, caso exista uma aplicação igual a decorrer no *smartphone* desse condutor/ocupante de imediato lhe aparecerá no mapa um ícone alusivo ao problema relatado por outro. Esse relatório de erro é também enviado através da Internet, caso exista acesso, para a base de dados remota para que possa ser visualizado por qualquer pessoa com acesso ao sistema.

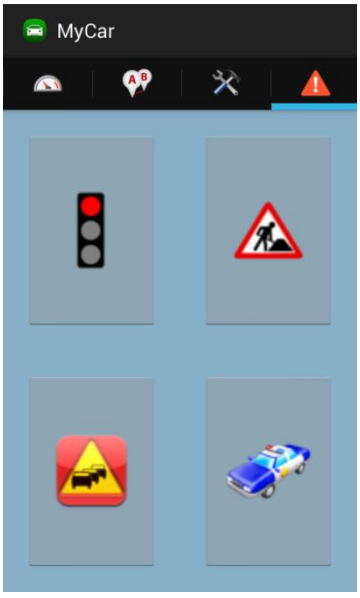


Figura 31 Ecrã com alertas de estrada para envio pelo utilizador.

Figura 32 Ecrã para seleccionar dados sobre o alerta de mapa para enviar via VANET e via Web.

Na figura 32, podemos observar o ecrã que surge ao utilizador, de forma semelhante ao que acontece no ecrã de envio de erros relacionados com o veículo (figura 30) para envio do alerta com mais detalhes fornecidos pelo utilizador antes do envio.

4.4 Sumário

Este capítulo que termina pretende mostrar como foi implementado o STRIVE a partir da ideia conceptual que existia.

Pretende-se mostrar como foi alterado o REINVENT de modo que desse suporte a vários tipos de mensagens além das que já existiam tornando-o mais abrangente e disponível para ser utilizado por utilizadores que pretendam desenvolver aplicações que enviam mensagens através da rede veicular.

Com estas alterações é possível através da instalação de uma aplicação, que funciona como *Content Provider* e permite enviar e receber diversos tipos de mensagens com alertas ou mensagens de texto enviados por utilizadores em redor, ou ainda dados de veículo enviados pela OBU que se liga ao veículo.

No capítulo ficou também demonstrado além desta camada de abstracção a comunicação na VANET, que é viável a criação de uma aplicação que faça uso dessa camada. Foi criada a aplicação *MyCar*, que mostra como a conjugação do envio de mensagens na rede veicular com serviços remotos através de Internet, podem gerar uma aplicação que se traduz numa ajuda à condução, gestão de veículo e entretenimento. A aplicação *MyCar* faz uso dos serviços que a API disponibiliza da mesma forma que qualquer outro utilizador pode fazer.

5. Testes

Neste capítulo irão ser apresentados alguns testes efectuados com o objectivo de mostrar a viabilidade do sistema através da demonstração da existência de comunicações entre todos os módulos do STRIVE que permitem que o utilizador disfrute de todas as funcionalidades da aplicação *MyCar* propostas.

Os testes desenvolvidos foram feitos em ambiente controlado, ou seja, não foi feito em circulação nas estradas pelo facto de o REINVENT já ter sido testado neste ambiente no trabalho em [9]. Contudo, todo o material necessário aos testes de comunicação entre elementos OBU, *smartphone*, serviços remotos foi usado em ambiente experimental. Existe apenas a necessidade de destacar a simulação de envio de dados de GPS e de sensores do veículo, nomeadamente combustível, por falta de elementos físicos para tal na OBU. Deste modo, garante-se a conectividade e funcionalidades necessárias para apenas uma adaptação para um cenário com componentes reais.

Vão ser apresentadas algumas imagens retiradas dos ecrãs dos *smartphones* com a aplicação em funcionamento, e imagens retiradas dos computadores ligados às OBUs usados para visualizar as diferentes mensagens que as placas recebem e enviam.

5.1 Material Usado

Foram usadas duas OBUs (com antenas Wi-Fi e WAVE) ligadas a dois computadores portáteis por cabo *Ethernet* recorrendo a uma sessão de SSH. As OBUs geram duas redes locais, sem acesso à Internet, onde se ligam dois *smartphones* Android, como ilustram as figuras 33A e 33B.

A **B**

Figura 33 Estações de teste utilizadas com computador, OBU e *smartphone*.

Para testar as ligações aos *web-services*, os *smartphones* foram ligados a uma rede, "eduroam" com acesso à Internet, pois as OBUs usadas não permitiam a ponte entre a interface *Ethernet* e Wi-Fi.

Todas as imagens que vão ser usadas nos testes foram captadas do *smartphone* e da linha de comandos da ligação da OBU, do sistema número 1.

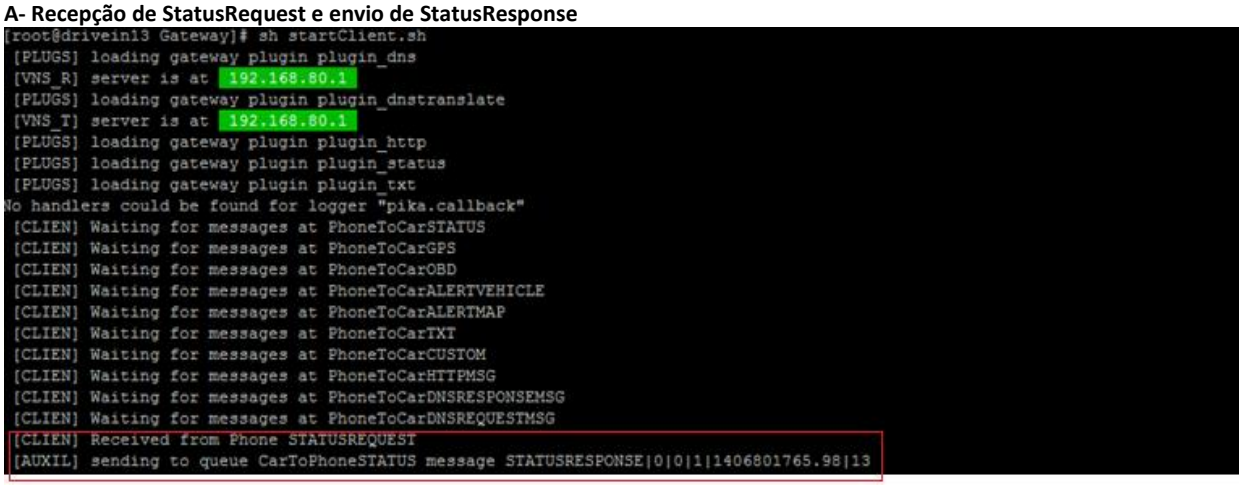
5.2 Testes efectuados

Tendo por base novamente o diagrama da figura 4, e os diagramas de sequência criados para os principais cenários, figuras 10, 13 e 15, foram efectuados alguns testes que pretendiam mostrar que as condições para realizar esses cenários foram atingidas. Prova também disso é a implementação de todas as funcionalidades propostas.

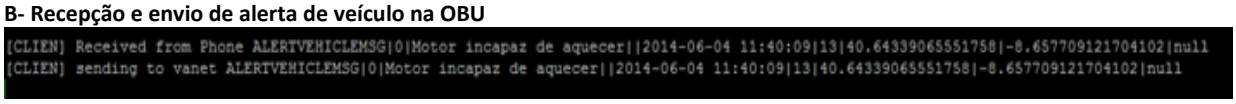
5.2.1 Conexão Smartphone com OBU

O primeiro teste efectuado foi o envio de mensagens de StatusRequest da aplicação para a OBU. É esperado que a placa retorne uma mensagem do tipo StatusResponse com alguns campos de informação para a aplicação.

A- Recepção de StatusRequest e envio de StatusResponse



B- Recepção e envio de alerta de veículo na OBU



C- Recepção e envio de mensagens de alerta de mapa do smartphone na OBU.

D- Recepção e envio de mensagens de texto entre 2 utilizadores.

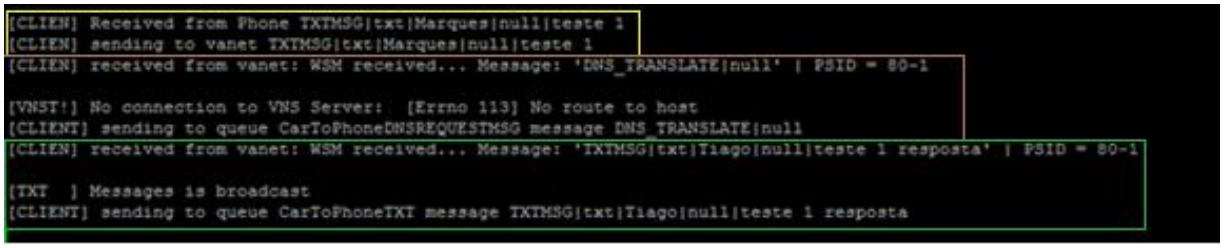


Figura 34 Imagens retiradas do computador 1 quando foi feito: A- envio de mensagens de *StatusRequest* e *StatusResponse* para teste de conectividade entre *smartphone e* OBU*;* B- envio de mensagens de alerta de veículo do *smartphone* para a OBU e para a VANET; C- envio de mensagens de alerta de mapa do *smartphone* para a OBU e para a VANET; D- recepção na OBU de mensagens de texto do *smartphone* e envio para a VANET e recepção de mensagem de texto da VANET e envio para *smartphone*.

Para ilustrar a comunicação entre OBU e Android nas próximas imagens temos exemplos de através de diferentes ecrãs da aplicação, do envio de diferentes mensagens para testar as ligações e as funcionalidades. Testamos as seguintes mensagens:

- Mensagem de alerta de veículo
- Mensagem de alerta de mapa
- Mensagem de texto entre veículos

Por exemplo, na figura 34A é possível ver-se a recepção da mensagem na OBU e o envio da resposta.

As mensagens de alerta de veículo (figura 34B) são visíveis na recepção da mensagem ALERTVEHICLE do telemóvel com os vários campos enviados pelo utilizador para a rede veicular.

Na segunda linha do texto da imagem está a indicação da OBU a informar que a mensagem segue para a rede veicular, naturalmente por *broadcast* pois é uma mensagem de aviso para todos os veículos.

As mensagens de alerta de mapa (figura 34C), de forma semelhante ao que se passa com o caso dos alertas de veículo, são recebidas pela OBU na interface de Wi-Fi como está visível na imagem, e são enviados para a VANET na interface WAVE.

As mensagens de texto entre veículos (na figura 34D) têm vários aspectos de relevância. A OBU recebe do *smartphone* uma mensagem de texto, com os campos "teste 1" (a mensagem em si), Marques (emissor da mensagem), null (receptor da mensagem).

Como primeira análise verifica-se, mais uma vez, no rectângulo amarelo, a correcta recepção de mensagens na OBU. Após recepção, envia essa mensagem para a rede veicular.

A informação que aparece na imagem relacionada com DNS, no rectângulo laranja, pertence a um módulo externo que foi desenvolvido em paralelo com este projecto para obter o ID da placa que se deve colocar no campo do receptor em vez do nome. Neste caso como foi colocado o "nome" Null no campo, o DNS não consegue traduzir para um ID específico, pelo que a mensagem vai em *broadcast* para a VANET. Este é um caso especial pré-estabelecido para *broadcast.* Quando não se pretende utilizar o DNS, no campo do receptor deve estar o ID da placa que irá receber a mensagem e passá-la

à aplicação. Deste modo fica limitado o envio de mensagens entre utilizadores, pois apenas funciona para um *smartphone* por veículo até que o DNS esteja completamente funcional.

5.2.2 Conexão OBU com OBU

Para testar a conectividade OBU-OBU, ou seja conectividade na VANET, foi enviada uma mensagem do outro *smartphone* que está ligado à OBU número 2, como se pode observar no rectângulo verde.

A mensagem é recebida pela OBU, desta vez vindo pela interface WAVE, o que mostra a conectividade entre os dois equipamentos do veículo.

Como já tinha sido referido anteriormente, as OBUs utilizadas não possuíam uma ligação ao OBD como alguns modelos mais recentes. Desta forma foi usado um simulador no REINVENT para enviar as mensagens para a aplicação com dados relativos ao veículo. Neste caso os dados simulados estão relacionados com o consumo, tipo de combustível e se o carro está desligado ou ligado. Na figura 35 pode-se observar o envio de mensagens da placa para o *smartphone* através das filas do RabbitMQ.

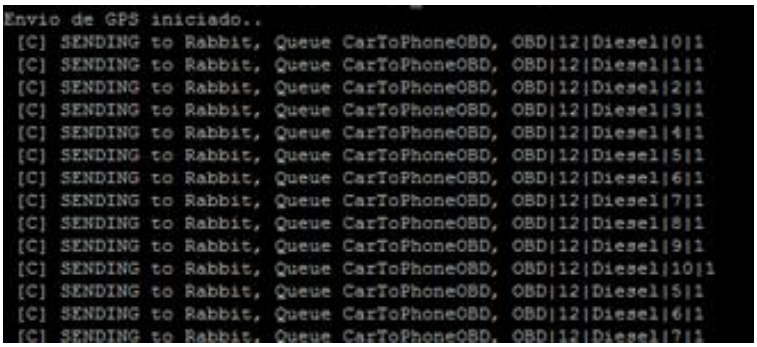


Figura 35 Envio de dados OBD simulados da OBU para *smartphone*

5.2.3 Conexão com servidor remoto

Nos testes foi também verificada a ligação entre a OBU e os serviços web. Verificou-se quer a inserção de dados da aplicação na base de dados remota, quer o fornecimento dos dados existentes na base de dados, à OBU e passagem dos mesmos à aplicação.

Para verificar que os dados enviados pela aplicação estavam a chegar de forma correcta ao servidor remoto, foi usado o *browser* do computador para visualizar os dados presentes nos diferentes *web services*. Como exemplo enviou-se na aplicação um alerta de veículo, através do ecrã visível na figura 30, que seguiu quer para a VANET quer para o servidor remoto. O alerta foi depois visualizado no *browser*, como se pode ver na figura 36, seguindo assim o diagrama de sequência previsto na figura 15.

```
[
  {
    "id":20,
    "carID":69,
    "timeStamp":"14/02/2014 19:30:25",
    "type":1,
    "problem":"Temperatura Elevada",
    "value":120.0,
    "description":"Semáforos",
    "latitude":10.1,
    "longitude":10.1
  },
  {
    "id":22,
    "carID":69,
    "timeStamp":"12/02/2014 19:30:25",
    "type":1,
    "problem":"Problema Oleo",
    "value":120.0,
    "description":"Semáforos",
    "latitude":10.1,
    "longitude":10.1
  }
]
```


Figura 36 Captura de ecrã retirada do *browser* na página do serviço remoto.

5.2.4 Gateway da VANET para HTTP

Esta *gateway* serve para poder reencaminhar alguns pedidos *HTTP* pela VANET quando a OBU não consegue encaminhá-los para a Internet directamente.

No teste efectuou-se um pedido simples de *HTTP* através da VANET de uma OBU para outra quando existia conectividade à Internet na OBU número 2 e não existia na OBU número 1.

Na figura 37 pode-se observar um exemplo de envio de mensagem com pedido *HTTP* do *smartphone* com destino à VANET, para outra OBU poder reencaminhar para a Internet. É visível na imagem, na primeira linha, a recepção na OBU da mensagem recebida na interface com a VANET. De seguida é visível a resposta que foi recebida pela VANET, com o código de erro para efeitos de *debug*.

Na última linha visível na imagem vê-se que a resposta foi enviada para a fila dedicada do RabbitMQ para fazer chegar a resposta ao *smartphone*. Neste caso foi usada uma fila de mensagens cujo destino é a VANET para que se faça chegar a resposta ao *smartphone* que fez o pedido.

```
[root@drivein12 Gateway]#  [CLIEN] received from vanet: WSM received... Message: 'HTTP|www.google.pt|GET|no body' | PSID = 80-1

[HTTP] response:  200 OK
[AUXIL] sending to queue PhoneToCarHTTPRESPONSE message HTTPRESPONSE|200|<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage
```

Figura 37 Recepção de mensagem HTTP da VANET e envio da resposta de volta para VANET para o *smartphone* que fez o pedido.

6. Discussão e conclusões

O objectivo deste trabalho era criar um sistema de partilha de informação de dados de trânsito e dos veiculos utilizando os *smartphones* e as redes veiculares. Pesquisas efectuadas mostravam a falta de ligação entre os *smartphones* e as redes veiculares, muito em parte devido à falta de compatibilidade entre os novos telefones e as redes/tecnologias de comunicação, como WAVE, existentes nas comunicações entre veículos.

Foi idealizado então uma solução que corrigisse esse problema para os *smartphones* com sistema operativo Android.

Assim surgiu o STRIVE. Um sistema que visa a partilha de dados de trafego com recurso a alertas e mensagens de texto entre utilizadores, sobre uma rede veicular.

O STRIVE foi desenvolvido tendo em vista 3 cenários principais: a comunicação entre *smartphone* e veículo; a comunicação entre aplicações dentro da VANET, a comunicação entre aplicações e recursos exteriores de suporte ao sistema criados no âmbito do STRIVE.

Para colmatar a falha identificada de falta de compatibilidade entre *smartphones* e VANETs foi adaptado o REINVENT [9], um sistema de troca de mensagens entre *smartphones* com sistema operativo Android e rede veicular, que apenas tinha suporte para mensagens do tipo texto e GPS.

Depois de compreendido o seu funcionamento foi extendida a sua capacidade para que permitisse a transmissão de outros novos tipos de mensagens incluindo diversos tipos de alertas para a criação de um sistema de transmissão de dados em tempo real que possa ser útil a um utilizador que seja condutor. O STRIVE funciona assim como uma API para a VANET para qualquer aplicação desenvolvida em Android que pretenda enviar ou receber mensagens num cenário de comunicação entre aplicações em redes veiculares.

Para servir de prova de conceito e de viabilidade do STRIVE, foi criada uma aplicação que, fazendo uso do sistema, trouxesse até aos utilizadores as funcionalidades que tinham sido enumeradas como fulcrais para o STRIVE representar uma ajuda na condução ou na gestão de veículo.

Essa aplicação de nome *MyCar* permite visualizar em tempo real alertas lançados por outros utilizadores através da VANET ou através de um acesso a um servidor remoto. Esses alertas irão aparecer sob a forma de avisos no mapa para fácil visualização por qualquer utilizador. A aplicação permite enviar diversos tipos de alertas que foram categorizados em 2 grupos (estrada ou veículo). Esses alertas são enviados para a VANET e para um servidor remoto caso exista conectividade com a Internet. Através do *MyCar* é possível ter acesso a dados transmitidos pelo veículo para a OBU através da interface OBD. Esses dados podem ser leituras do consumo instantâneo ou alertas de problemas mecânicos detectados pelo sistema electrónico do carro que iram lançar um alerta no aplicativo no utilizador.

Além destas funções, o utilizador pode utilizar a aplicação para fazer uma gestão de rotas por onde o veículo tem circulado, com informação detalhada de localização e consumos instaneos e/ou médios,

ou ainda para trocar mensagens de texto com outros utilizadores, o que atribui uma característica de entretenimento à aplicação.

Desta forma a aplicação consegue envolver todos os cenários que tinham sido identificados verificando-se as funcionalidades que o STRIVE disponibiliza.

O sistema depois de desenvolvido e implementado foi testado utilizando componentes (OBU e *smartphones*) reais que permitiram concluir a viabilidade do mesmo em cenário real de funcionamento, conseguindo atingir as funcionalidades pensadas para os utilizadores que são, em alguns cenários, os condutores dos veículos.

Por fim concluímos que o STRIVE é um sistema viável passível de ser implementado e usado em qualquer cidade que possua infra-estrutura que o suporte, trazendo claras vantagens a todos os que circulam pelas estradas num veículo ligado a uma VANET, utilizando com interface a aplicação *MyCar*. Devido a sua arquitectura especialmente pensada para tal, serve ainda como incentivo ao desenvolvimento de novas aplicações e/ou sistemas que explorem as redes veiculares como via de comunicação e o poder computacional dos *smartphones*.

Como ultima análise conclui-se que foi um projecto bem conseguido com resultado positivo.

6.2 Trabalho futuro

O STRIVE permite ser uma base para a abstracção das redes veiculares, mas existem várias melhorias a realizar ao sistema e também na aplicação MyCar, que são apresentadas de seguida:

- A nível da comunicação, deve ser alterada a configuração do gestor de mensagens RabbitMQ para que permita a utilização de vários *smartphones* ligados em simultâneo a cada OBU.
- Na aplicação seria interessante se se adicionassem comandos por voz à aplicação, diminuindo a necessidade de interacção por toque do condutor/utilizador.
- Melhorar o funcionamento da obtenção de rotas antigas na aplicação de modo a ser mais eficiente.
- Melhorar o funcionamento de obtenção de dados do veículo e a forma como são mostrados os erros ao utilizador.
- Melhorar os *layouts*/ecrãs da aplicação. Os ecrãs devem sofrer algumas alterações de modo a tornar a interacção com a aplicação o mais simples e agradável possível

7. Referências

[1] S. Zeadally, R. Hunt, Yuh-Shyan Chen, A. Irwin,A. Hassan, "Vehicular ad hoc networks (VANETS): status, results,and challenges, 2010.

[2] John B. Kenney, "Dedicated Short-Range Communications (DSRC) Standards in the United States" Vol. 99, No. 7, July 2011.

[3] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions", June 2009.

[4] C. Merlin and W. Heinzelman, "A study of safety applications in vehicular networks," IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, pp. 1–8, 2005.

[5] Elmar Schoch, Frank Kargl, and Michael Weber, "Communication Patterns in VANETs", November 2008.

[6] CISCO, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018", February 2014

[7] Statista iOS: http://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/, data de ultimo acesso 17/06/2014.

[8] Statista Android: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/, data de ultimo acesso 17/06/2014.

[9] Filipe Oliveira, Susana Sargento, José Maria Fernandes, André Cardote, "REINVENT: Accessing Vehicular Networks in Mobile Applications", Dissertação de Mestrado, Universidade de Aveiro, 2013.

[10] TechTerms- "Ad Hoc Network": http://www.techterms.com/definition/adhocnetwork, data de ultimo acesso 17/06/2014.

[11] S. Corson, J. Macker "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations" January 1999.

[12] H.Krishnan, "Vehicle Safety Communications Project", February 2006.

[13] Internet Systems Consortium: https://www.isc.org/mission/, data de ultimo acesso 17/06/2014.

[14] PReVENT: http://www.esafetysupport.info/download/research_and_development/PReVENT_IP_Presentation_v20.pdf,data acesso 17/06/2014, data de ultimo acesso 17/06/2014.

[15] IEEE 802.11p : http://www.itwissen.info/definition/lexikon/802-11p-IEEE-802-11p.html, data de ultimo acesso 17/06/2014.

[16] S. Biswas, R. Tatchikou, F.Dion, "Vehicle-to-Vehicle Wireless Communication Protocols for Enhancing Higway Traffic Safetty", January 2006.

[17] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, T Weil “Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions”, 2010.

[18] S.Yousefi, M. Mousavi, M. Fathy “Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives”, 6th International Conference on ITS Telecommunications Proceedings, 2006.

[19]J. Ott, D. Kutscher “Drive-thru Internet: IEEE 802.11b for “Automobile” Users”, 2004.

[20]J. Singh, N. Bambos, B.Srinivasan, D.Clawin “ Wireless LAN Performance Under Varied Stress Conditions in Vehicular Traffic Scenarios”, 2002.

[21] Unai Hernandez, Asier Perallos, Nekane Sainz, and Ignacio Angulo “Vehicle On Board Platform: Communications Test and Prototyping”, 2010 IEEE Intelligent Vehicles Symposium, June 2010, pp. 967-972.

[22]National Highway Traffic Safety Administration: http://www.nhtsa.gov/About+NHTSA/Press+Releases/2014/USDOT+to+Move+Forward+with+Vehicle-to-Vehicle+Communication+Technology+for+Light+Vehicles , data de ultimo acesso 17/06/2014.

[23] S. Rizvi, S. Olariu, C. Pinotti, S. Salleh,M. Rizvi, Z. Zaidi “Vehicular Ad Hoc Networks”, International Journal of Vehicular Technology,Vol. 2011, Article ID 256542, 2 pages.

[24] Hindawi, International Journal of Vehicular Technology: http://www.hindawi.com/journals/ijvt/2011/256542/, data de ultimo acesso 17/06/2014.

[25] M. Esbjörnsson, O. Juhlin, M. Östergren “Making Motor Bikers Come Together - Fast Moving Users and Mobile Ad Hoc Networks”, Mobility, Interactive Institute, P.O. Box 24081, SE – 104 50 Stockholm, Sweden.

[26] G. Macario, M. Torchiano, M. Violante “An In-Vehicle Infotainment Software Architecture Based on Google Android”, IEEE 2009.

[27] Apple, “CarPlay”: http://www.apple.com/ios/carplay/, data de ultimo acesso 17/06/2014.

[28] TIME, http://time.com/11407/volvo-shows-off-the-apple-carplay-iphone-interface/, data de ultimo acesso 17/06/2014.

[29] J.Jeyanthy Lizy, M.Varghese “A High Secure And Efficient Data Acquistion Mechanism In Vehicular Ad Hoc Networks (Vanets)” IOSR Journal of Electronics and Communication Engineering (IOSR-JECE),Vol 9, Issue 1, Ver. VI (Feb. 2014), PP 93-98.

[30] RabbitMQ: http://www.rabbitmq.com/, data de ultimo acesso 17/06/2014.

[31] Gson: https://code.google.com/p/google-gson/, data de ultimo acesso 17/06/2014.

[32] Persistence: http://docs.oracle.com/javaee/7/api/javax/persistence/package-summary.html, data de ultimo acesso 17/06/2014.

[33] S. Diewald, A. Möller, L. Roalter, and M. Kranz, “DriveAssist-A V2X-Based Driver Assistance System for Android.,” Mensch & Computer, 2012.

[34] J.Jeyanthy Lizy, M.Varghese “A High Secure And Efficient Data Acquistion Mechanism In

Vehicular Ad Hoc Networks(Vanets), IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 9, Issue 1, Ver. VI (Feb. 2014), PP 93-98.

[35] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture," ACM Transactions on Internet Technology, vol. 2, no. 2, pp. 115–150, May 2002.

ANEXO A - Detalhe dos principais métodos da classe NetworkProvider.java

- Recepção de mensagens de mensagens do *provider* para a aplicação.

- *call(String URI, String method, String args, Bundle extras) :*

Este método é utilizado pela aplicação quando quer obter uma mensagem nova, imediatamente a seguir a obter uma notificação lançada pela classe *MessageListenerService.java*.

- No campo URI, deve estar no URI para *provider* que deve ser utilizado para obter a informação.

- No campo *method*, deve estar um identificador do tipo de recurso (mensagem) que queremos obter do *provider*. Os métodos podem ser encontrados na classe auxiliar para que o utilizador da API não tenha de decorar os mesmos.

- No campo args, deve ser colocada uma *string* que serve para do lado do *provider* fazer uma pesquisa mais rápida do tipo de mensagem pretendida. Faz sentido num contexto de ter mais de um subtipo de mensagens associadas ao mesmo recurso. Este modelo foi idealizado, mas não chegou a ser completamente implementado pelo que aceita se a *string* for "vazia", mas não deve ir com valor *Null*.

- O campo *extras*, tem o valor *Null*.

- Como retorno, a função devolve um *Bundle*, usado para passar informações dentro de aplicações, e de aplicações para outras aplicações. Dentro do *Bundle*, a informação está guardada numa estrutura de chave – objecto. Ou seja, para obter a informação temos de saber o nome exacto da chave com que foi guardado. Para facilitar esse processo, dentro da classe *StaticUtils.java* encontram-se as chaves para obter as mensagens do *bundle* recebido, tornando deste modo o *provider* mais protegido, mas de fácil acesso.

- Envio de mensagem da aplicação para *provider*.

- *insert(URI, Content Values values):*

Este método é utilizado pela aplicação quando quer enviar dados para o *provider* para serem enviados para a VANET. Este método vai ficar encarregue de receber os dados da aplicação, construir as mensagens dos vários tipos de forma conveniente, e proceder no fim ao seu envio para a rede veicular.

- Campo *URI*: No campo URI deve estar o *path* completo para o recurso (mensagem) que queremos enviar. Deste modo o *provider* sabe com que tipo de mensagens deve "lidar".

- Campo *values*: Em ANDROID uma possível maneira de enviar dados entre aplicações é recorrendo a um *Content Values*, cujo funcionamento e conceito é semelhante ao *Bundle* ,mas mais indicado quando se pretende fazer um *insert* de dados num *provider*.

ANEXO B – Tipos de mensagens suportados na VANET

Nota: As mensagens relacionadas com DNS não foram descritas em maior pormenor por não fazerem parte deste trabalho.

Mensagem de alertas relacionados com o mapa

Figura 38 Formato das mensagens de alerta de mapas

- **Tipo**: Campo que identifica o tipo de alerta que se quer transmitir. Os valores estão definidos por defeito para o tipo de erros implementados.
- **Problema**: Campo que identifica o problema relatado.
- **Descrição**: Campo opcional que adiciona informação ao problema relatado.
- **TimeStamp**: Campo que identifica o instante em que a mensagem foi enviada.
- **IDEmissor**: Campo que identifica o veículo emissor da mensagem.
- **Latitude**: Campo que informa da latitude do ponto onde o problema ocorreu.
- **Longitude:** Campo que informa da longitude do ponto onde o problema ocorreu.

Mensagem de alertas relacionados com o veículo:

Figura 39 Formato das mensagens de alerta de veículo

- **Tipo**: Campo que identifica o tipo de alerta que se quer transmitir. Os valores estão definidos por defeito para o tipo de erros implementados.
- **Problema**: Campo que identifica o problema relatado.
- **Descrição**: Campo opcional que adiciona informação ao problema relatado.
- **TimeStamp**: Campo que identifica o instante em que a mensagem foi enviada.
- **IDEmissor**: Campo que identifica o veículo emissor da mensagem.
- **Latitude**: Campo que informa da latitude do ponto onde o problema ocorreu.
- **Longitude**: Campo que informa da longitude do ponto onde o problema ocorreu.

Mensagem de texto:

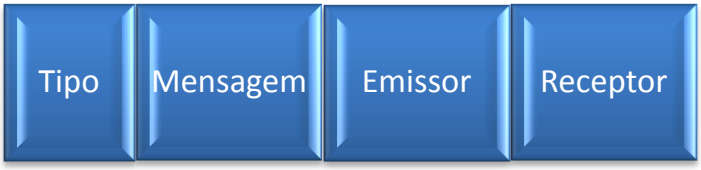


Figura 40 Formato das mensagens de texto

- **Tipo:** Campo que identifica o tipo de alerta que se quer transmitir. Os valores estão definidos por defeito para o tipo de erros implementados.
- **Mensagem:** Campo com o corpo da mensagem.
- **Emissor:** Campo com o identificador do emissor.
- **Receptor:** Campo com o identificador do destinatário.

Mensagem de StatusRequest:

Figura 41 Formato das mensagens de pedido de estado

- **Tipo:** Campo que identifica o tipo de mensagem como StatusRequest.

Mensagem de StatusResponse:

Figura 42 Formato das mensagens de resposta a estado

- **Tipo**: Campo que identifica o tipo de mensagem como resposta a um StatusRequest ou seja um StatusResponse.
- ***Wi-Fi***: Campo que indica se a placa do carro (OBU) está ligada a uma rede *Wi-Fi*.
- **Vanet**: Campo que indica se a placa do carro (OBU) está ligada a uma rede veicular (WAVE).
- **Celular**: Campo que indica se a placa do carro (OBU) está ligada a uma rede celular (3G).
- **TimeStamp**: Campo que indica quando foi feito o envio da mensagem.
- **Emissor**: Campo que indica o ID da placa do veículo.

Mensagem de Http:

Figura 43 Formato das mensagens de pedidos HTTP

- **Tipo**: Campo que identifica o tipo de alerta que se quer transmitir. Neste caso o tipo será um identificador de HTTP.
- **Emissor**: Campo que identifica a placa que enviou esse pedido para a VANET.
- **URL**: Campo com o url destino do pedido.
- **Método**: Campo com o método que queremos usar (exemplo GET, PUT,POST).

Mensagem de HttpResponse:

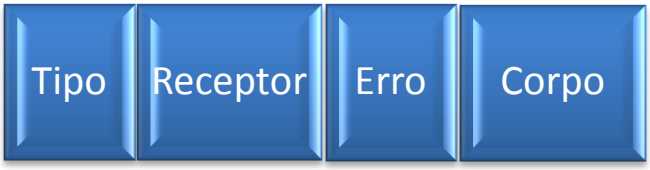


Figura 44 Formato das mensagens de resposta httpHTTP

- **Tipo**: Campo que identifica o tipo de alerta que se quer transmitir. Neste caso o tipo será um identificador de resposta a uma mensagem http.
- **Receptor**: Campo que identifica a placa destinatária da mensagem.
- **Erro**: Campo com o erro dado pelo servidor (exemplo 200 OK).
- **Corpo**: Campo com o método que queremos usar (exemplo GET, PUT,POST).

Mensagem de GPS:

Figura 45 Formato das mensagens GPS

- **Tipo**: Campo que identifica o tipo de alerta que se quer transmitir. Neste caso o tipo será um identificador de resposta a uma mensagem http.
- **Latitude**: Campo que indica a latitude do ponto geográfico actual da OBU.
- **Longitude**: Campo que indica a longitude do ponto geográfico actual da OBU.
- **Altitude**: Campo que indica a altitude do ponto geográfico actual da OBU.
- **Velocidade**: Campo que indica a velocidade de circulação do veículo.
- **Precisão**: Campo que indica precisão dos dados de posição geográficos.
- **TimeStamp**: Campo que indica o instante de envio da mensagem.

Mensagem de OBD:

Figura 46 Formato das mensagens OBD

- **Tipo**: Campo que identifica o tipo de alerta que se quer transmitir. Neste caso o tipo será um identificador de mensagem OBD.
- **Emissor**: Campo que identifica a placa emissora da mensagem.
- **Combustível**: Campo com o tipo de combustível do veículo.
- **Consumo**: Campo com valor de consumo do veículo.
- **Estado**: Campo com identificador se veículo está ligado ou desligado.

Mensagem de Utilizador:

Figura 47 Formato das mensagens de utilizador da API

- **Tipo**: Campo que identifica o tipo de alerta que se quer transmitir. Neste caso o tipo será um identificador de uma mensagem de utilizador.
- **Código**: Campo com o código dado pelo utilizador para identificar o tipo de mensagem.
- **Corpo**: Campo com a mensagem que se quer enviar. A gestão da divisão do corpo da mensagem em partes cabe ao utilizador da API.